Modelling the spread of waterborne disease in salmon farms by means of a Lagrangian particle tracking model

by

Meghan Kennealy

Thesis presented in partial fulfilment of the requirements for the degree of Master of Science (Applied Mathematics) in the Faculty of Science at Stellenbosch University

Supervisor: Prof GJF Smit
Co-supervisor: Dr JM Wilms

December 2020

The financial assistance of the Centre of Excellence of Mathematical and Statistical Sciences (CoE-MaSS) and National Research Foundation (NRF) towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at are those of the author and are not necessarily to be attributed to the NRF.

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: July 2020 .....

Abstract

This study sets out to model the spread of a waterborne disease between cages within an open water salmon farm, by means of a Lagrangian particle tracking model. The study modelled the flow through and around an array of salmon fish cages in an open ocean environment by means of computational fluid dynamics (CFD). Throughout the study an in-house code, which is a Python package called Fish Infection Simulation Helper (FISH), was developed. The code FISH was developed to simulate the spread of virus particles. The particles are generated within a initially infected region with a population model. The particles are then tracked throughout the domain by means of a Lagrangian particle tracking model. FISH was used as a post-processing tool which was coupled with the OpenFOAM CFD model of the velocity field. The disease model was comprised of a population model as well as a shedding and decay model to account for the behaviour of the virus particles.

Opsomming

Hierdie studie het ten doel om die verspreiding van 'n watergedraagde siekte tussen hokke in 'n oopwater-salmplaas deur middel van 'n Lagrangiaanse deeltjie-opsporingsmodel, te modelleer. Die studie het die vloei deur en rondom 'n verskeidenheid salmvishokke in 'n oop oseaan-omgewing deur middel van berekeningsvloeidinamika ("CFD"), gemodelleer. Gedurende die studie van die inhuis kode is 'n Python-pakket genaamd Fish Infection Simulation Helper (FISH) ontwikkel. Die kode FISH is ontwikkel om die verspreiding van virus-deeltjies te simuleer namate die deeltjies deur die hele domein gegenereer word, deur 'n bevolkingsmodel te kombineer met 'n Lagrangiaanse deeltjie-opsporingsmodel. FISH is gebruik as 'n naverwerkingsprogram wat gekoppel is aan die OpenFOAM CFD-model van die snelheidsveld. Die siekte-model bestaan uit 'n bevolkingsmodel sowel as 'n stortings- en vervalmodel om die gedrag van die virusdeeltjies te modelleer.

Acknowledgements

This work would not have been possible without the financial support of the DST-NRF Centre of Excellence in Mathematical and Statistical Sciences (CoE-MaSS).

I would like to express my appreciation and sincerest thanks towards Prof Smit for his constant support, encouragement, compassion and guidance throughout this thesis. I am so grateful to have had the opportunity to work with someone who believed in me unconditionally and pushed me to continuously improve.

I would also like to express my gratitude to Dr Wilms for keeping me sane and for her constant mentorship. Thank you for enduring my questions at all hours, when Google just would not suffice.

Last but not least, I would like to thank Lucy the dog for making sure I took regular breaks to scratch her tummy.

Dedications

To the memory of my mother.

Contents

List of Figures

List of Tables

Nomenclature

Variables

A	Area of the faces of a cell
C_n, D_n	Normal resistance coefficients
C_t, D_t	Tangential resistance coefficients
dr	Infinitesimal changes in fractional position
ds	Infinitesimal changes in pseudo time
dt	Infinitesimal changes in time
du, dv	Infinitesimal changes in velocity
dW	Infinitesimal changes in the Wiener process
dx, dy, dz	Infinitesimal changes in distance
e_1, e_2, e_3	directional scaling factors
E	Exposed population
F, G, H	Transports in the x, y, z directions
h, l	simplifications for equation solving
I	Infection population
K_H	Horizontal diffusion coefficient
K_V	Vertical diffusion coefficient
m, n	Fractional position within the cell
N	Total fish population size
N_x, N_y	Grid size in the x and y directions

P	Pressure
q_1, q_2, q_3, q_4	Runge Kutta coefficients
Q	Volumetric flow rate
Q_s	Volumetric rate of production
r	Fractional position within the cell
R	Removed population
s	Pseudo time
S	Susceptible population
t	Time
T	Total time
u	Velocity in the x direction
U	Velocity field in the x direction
U_0	Incoming velocity
v	Velocity in the y direction
V	Velocity field in the y direction
w	Velocity in the z direction
W	Velocity field in the z direction
W_i	Coefficient of Wiener process matrix
x	Position in the x direction
x_1, x_2	Faces of a cell in the x direction
y	Position in the y direction
y_1, y_2	Faces of a cell in the y direction
z	Position in the z direction
z_1, z_2	Faces of a cell in the z direction
$\mathcal{C}$	Courant number
$\mathcal{D}$	Diffusion coefficient

$\mathcal{V}$	Volume of a cell
$\dot{\mathcal{V}}$	Volume rate of water created or consumed
$\mathcal{W}$	Waterphase
$\mathcal{Z}$	Elements of independent random numbers vector
α_i	Components of the temporal velocity gradient
β	Transmission rate
γ	Removal rate
Γ	Shedding rate
ΔF	Change in transport
Δs	Change in pseudo time
Δt	Timestep
$\Delta U, \Delta V$	Change in U and V velocity fields
$\Delta x, \Delta y, \Delta z$	Change in the x, y, z direction
$\Delta\mathcal{V}$	Volume in a cell, $\Delta\mathcal{V} = \Delta x \Delta y \Delta z$
ζ	Elements of independent random number vector
λ	Decay rate
μ	Birth rate
ν	Death rate
σ	Infection rate
ϕ	Porosity
φ	Minimum infection threshold

Matrices and Vectors

$\mathbf{a}$	Deterministic advection term vector
$\mathbf{B}$	Diagonal matrix of diffusion coefficients
$\mathbf{C}, \mathbf{D}$	Diagonal matrix of resistance coefficients
$\mathbf{p}$	Particle position as a vector
$\mathbf{v}$	Velocity vector
$d\mathbf{W}$	The Wiener vector, $\boldsymbol{\zeta}\sqrt{dt}$
$\mathbf{x}$	Position vector
$\boldsymbol{\mathcal{Z}}$	Vector of independent random numbers
$\boldsymbol{\alpha}$	Temporal velocity gradient vector
$\boldsymbol{\zeta}$	Vector of independent random numbers

Subscripts and Superscripts

0	Initial timestep
$1, 2$	Indicates face
e	Exiting position of the particle
i	Indicates direction, where $i = x, y, z$
k	Current time step
N_x, N_y	Indicates the value in the last position in the grid
x, y, z	Indicates direction

Chapter 1

Introduction

Fish farming is increasingly important as the demand on food supplies grows with the world's population. The United Nations [2019] predicted that the world's population will reach 9.7 billion people by 2050. The study by the Global Salmon Initiative [2020a] outlined the efficiency and environmental impact of salmon farming versus other animal protein. They found that salmon is the most efficient source of animal protein, as the carbon footprint is almost one tenth that of beef, and the percentage yield is 40% higher than that of lamb, more statistics can be found in Appendix A. As salmon farming is an efficient way to feed the population, it is important to maximise the output by minimising the spread of disease within farms.

In 2009, 50% of salmon on the market was farmed, in 2014, the percentage had risen to 70% and by 2017 the percentage of farmed salmon was 75% [Howard, 2014; Live Science Staff, 2009; Trilling, 2017]. It is clear that the industry is growing rapidly, and with the rapid rise in farming, the possibility of disease spread increases. Limiting the spread of disease within such farms is vital, as the detection of such diseases calls for immediate eradication of the infected farm.

Fish infected with infectious salmon anaemia virus (ISAV), experience a loss of appetite, grey gills, swollen abdomen, swollen liver, kidney and spleen, among other symptoms [Canadian Food Inspection Agency (CFIA), 2020]. The mortality rate of fish infected with ISAV can be as high as above 90% [Spickler, 2010], and there is currently no treatment, cure or vaccine to lessen the effect of the disease on the fish. Although ISAV is not dangerous to humans, if the disease is detected in a farm, it must be eradicated as it is extremely destructive within the fish population.

The destruction is not limited to farmed fish and can have an impact on the wild population and subsequently the environment. Outbreaks of ISAV can also lead to transmission to the wild fish populations, threatening the natural ecosystem of the fjords in which the farms are located.

The farming cycle of the Atlantic Salmon lasts three years. The fish are initially raised in fresh water and transferred to the open water salmon farms after a year. The fish remain in the open water farms for another two years [Global Salmon Initiative, 2020b]. As required by law, the fish must be eradicated if the disease, ISAV, is detected. As the farming cycle is three years, the eradication of a farm can lead to a wastage of resources [Salama and Murray, 2013].

In August 2017, a breakout in Bomlo, Norway, led to the destruction of 900 000 fish that could not be harvested and sold as consumable protein. The retail value of a kilogram of Norwegian salmon meat was 5.96 EUR in August 2017 [Fish Pool Index, 2017]. The average weight of a mature Atlantic Salmon is 4.5 kg [National Oceanic and Atmospheric Administration (NOAA), 2019], and the edible yield of the fish is around 68% [Global Salmon Initiative, 2020a]. The destruction of an infected farm of 900 000 fish is thus around 16.4 million EUR, or 17.7 million USD in lost revenue for the farm, as well as three years of production time. These farms are traditionally in areas such as Chile, Norway, Canada, and Scotland; and more recently in Australia, Faroe Islands, Iceland, Ireland and New Zealand [Global Salmon Initiative, 2020b].

Optimal cage placement from the prediction of disease spread could potentially limit the spread of infections within these farms. The limitation of disease is best done with accurate models.

1.1 Background

As was discussed previously, the industry is growing rapidly. Eradicating a farm is a loss of revenue, time and vital protein. It is therefore in the best interest to limit the spread of disease to a single farm or, if possible, even to a single cage within a farm to ensure minimal resource losses. The velocity fields used to predict the spread of these diseases should be as accurate as possible as the virus particles are carried by the flow.

The problem of virus spread between salmon farms, where farms consist of multiple cages, has been considered in a number of previous studies. This section will review some previous work done to predict the transmission of viruses between salmon farms, as well as previous attempts at modelling the flow through and around cages in a velocity field. This chapter will also review other modelling techniques that may refine the model to be applied to cages within a farm.

Waterborne disease spread

Previous attempts at modelling the spread of disease have been on the spread between farms. This study will apply these methods to a smaller scale within

the farm as opposed to between multiple farms. Salama and Murray [2011] considered the size of the salmon farm as a factor in the transmission of pathogens between two farms and the distance the disease has travelled while the concentration of the pathogens is above the minimum infectious threshold. The minimum infection threshold is the minimum number of virus particles (virions) in an area where the infection will spread to another fish in that area.

Salama and Murray [2011] considered three diseases, two viral and one bacterial. The viral infections were infectious pancreatic necrosis virus type (IPNV-type) and infectious salmon anaemia virus type (ISAV-type). The bacterial infection was Aeromonas salmonicida bacteria type (AS-type). Salama and Murray [2011] used the susceptible, exposed, infected and recovered population model, discussed in Chapter 2. The Susceptible Exposed Infected Recovered model (SEIR) was used to model the infection within the fish population and the spread of disease from the infected farm to the downstream farms. The transmission between farms is done by means of a waterphase. Once the infection reached the second farm, by means of the waterphase, Salama and Murray [2011] made use of a population model for the second farm with a risk probability that informed the population model within the second farm. This links the population models of the initially infected farm and the second farm downstream. The paper written by Salama and Murray [2011] is unclear on what initial values are used for the susceptible, exposed and infected populations in the disease model, which has an impact on how the the infection spreads and how many fish are infected. It is suspected that urine is responsible for the initial spread of the disease in the early stages of a ISAV outbreak [Bricknell, 2017].

Salama and Murray [2011] found that the optimal separation distance was dependent on farm size, i.e. larger farms require larger separation distances. Viruses with higher decay rates allow for smaller separation distances between farms, as virions (virus particles) become inefficient sooner than those with longer decay times. The separation distances evaluated in Salama and Murray [2011] were between 1 km and 10 km. Salama and Murray [2011] noted that environmental factors such as ultra violet exposure, salinity and temperature can influence the decay of the virions, this would indicate that the rate of decay of a virus is not constant, however, without experimental data it is uncertain to what extent these factors influence the rate of decay. Salama and Murray [2011] only considered the role that farmed fish have in disease transmission and not wild fish or escaped fish. This role will have to be investigated further to determine the significance. Fish in the wild are complex to consider as it would involve evaluating the frequency in which they come in contact with the cages and whether or not the time the fish spends in such proximity is sufficient for the fish to be infected with the virus.

The study by Salama and Murray [2013] is similar to Salama and Murray

[2011], however the study did not include a population model for the second farm and focused on the hydrodynamic model and the role that the site biomass plays in transmission. Salama and Murray [2013] made a comparison between a computational modelling procedure and an analytical expression model to determine the mean transmission distances of the infection. They did not compare the computational results with experimental results, which makes this model difficult to assess, as it only compared results to that of the analytical model in Salama and Murray [2011]. Salama and Murray [2013] also considered the possibility of constructing the problem in a spreadsheet, which could allow for farm owners and authorities to calculate the minimum spacing between farms. This approach is limited by the calculations of the disturbance on the velocity field caused by the presence of obstacles, such as the fish farm cages, although it is possible that these could be done in a spreadsheet if the velocity fields used are based on data from the sites in question. As the cage dynamics have a significant effect on the downstream velocity fields, these calculations should not be disregarded, even though the study looks at the transmission between farms and not cages within a farm. The model presented by Salama and Murray [2013] is biologically and physically simplistic which results in a model that is computationally inexpensive. The physical attributes of the virus are taken to be homogeneous and the effects of bathymetry, topography and obstacles are not mentioned within the study. Although this study is simplistic, Salama and Murray [2013] can be used to attain worst case transmission distances between farms.

This is also the case with Salama and Murray [2011], which states that although the study is limited by simplicity, the simplification provides accurate dispersal models. Salama and Murray [2013] concluded that susceptibility to infection was not influenced by farm size, but it did play a role in the persistence of an infection, as was found in Salama and Murray [2011].

Population models

A population model is considered to model the number of virions that are active within the system. Allen [1994] considered the differences between the SI, SIR and SIS epidemic models, where S is the susceptible population, I is the infected population and R is the recovered population. It considered the link between the discrete-time models and the continuous models. Allen [1994] focused on chaotic behaviour in the models, which does not concern our population models, as it is only relevant in the case of the SIS models or in the SIR model with vital dynamics. The vital dynamics refers to the birth and death rates and the effect that rates have on the population model.

The Institute for Disease Modeling [2019] focused on population models where the disease has an incubation period. The overview given by the Institute for Disease Modeling [2019] described the dynamics of both the continuous SEIR

and SEIRS models with and without vital dynamics. The Institute for Disease Modeling [2019] gave insight into how the populations interact and touched on the differences between the SEIR and SEIRS models.

Li et al. [1999] considered the SEIR model where hosts are in direct contact with uninfected fish. Li et al. [1999] also considered a parameter that gave insight into whether a disease infects the majority of the population, or whether the disease dies out as a function of population size. Neither the Institute for Disease Modeling [2019], Allen [1994] nor Li et al. [1999] consider how the initial populations of the susceptible, exposed and infected are determined in practice.

Salama and Murray [2011] and Salama and Murray [2013] both made use of the SEIR model to represent the dynamics of the infectious salmon anaemia virus, although neither study gave insight into how the initial conditions of the model was chosen.

Qviller et al. [2020] suggested an alternative approach that considers a transmission model specifically for waterborne diseases in salmon farms. These are not neighbouring cages, but farms located significantly farther than that which will be considered in this study. The model however did not consider the flow field, and took a statistical approach to modelling the spread of disease from infected farms to susceptible farms based on the time delay before eradication and the density and proximity of surrounding susceptible farms. The study focused on the optimal time to destroy infected farms and concludes that the sooner a farm is eradicated, the smaller the probability that the disease will spread. The model presented in Qviller et al. [2020] will not be considered as it is outside of the scope of this study.

Particle tracking

Once the number of particles (virions) in circulation is known, the Lagrangian particle tracking model with a diffusion term is necessary to simulate the particle dispersion within the current. This section considers previous algorithms that were developed to simulate the motion of a particle that follows the path of the fluid in which it is suspended. Pollock [2016] mapped out the process to create a particle tracking algorithm. The guide constructed a particle tracking algorithm from a fluid element. The velocity is calculated by means of linear interpolation. The particle's exit position in the cell is calculated for each cell, which implies that cells cannot be skipped, although the Pollock [2016] algorithm is adaptable to various grid sizes, it may be computationally expensive for finer grids, and inaccurate for large grids.

Other particle tracking algorithms [Beletsky et al., 2007; Blanke and Raynaud, 1997; Fabbroni, 2009] took the timestep into consideration and calculated the particle's position at that timestep, whereas the Pollock [2016] algorithm com-

putes the exit time and exit location of the particle in a cell and repeats this until a boundary cell is reached, or the simulation time is reached. The intermediate particle locations are not computed. The algorithm gives a semi-analytical solution, as it iteratively calculates an analytical solution. The Pollock [2016] algorithm only gives insight into the particle's position on the grid points. The Pollock [2016] algorithm assumes that all velocities are nonzero and in the positive directions, i.e. there is no allowance for negative velocities within the algorithm. The algorithm created by Pollock [2016] therefore does not adapt to irregular flow fields, without possibly adapting the current algorithm, as is discussed in Section 3.3.1. A simpler solution would, however, be to consider models that adapt to irregularities without adapting the algorithm.

Fabbroni [2009] considered both particle tracking algorithms and diffusion. The Lagrangian model in Fabbroni [2009] tracks passive tracers in the open ocean that follow the same path as a fluid element. They made use of a Brownian motion model, a random walk model and, sub-grid diffusion to create a particle tracking model that simulates the path taken by a fluid element. Fabbroni [2009] considered the ARIANE particle tracking algorithm by Blanke and Raynaud [1997], the TRACE particle tracking algorithm developed by Beletsky et al. [2007], and SINCEM and SINCEM2, both developed at the Laboratorio di Simulazione Numeriche del Clima e degli Ecosistemi Marini. Fabbroni [2009] used the inertial oscillations model[1] as well as the Stommel solution[2] to validate the four particle tracking algorithms. They stated that SINCEM and SINCEM2 were also found to have a linear relationship between the Lagrangian timestep and the absolute error, a lower timestep was preferred by both algorithms. This study concluded that the predictability of the Lagrangian models is strongly dependent on the temporal resolution of the Eulerian velocity fields, the higher time frequency of the Eulerian velocity fields gave a better result. Fabbroni [2009] found, by means of computational experimentation, that the hourly and three hourly fields gave a better result than the daily mean current. Fabbroni [2009] also found the SINCEM2 with Runge-Kutta and SINCEM to be the most accurate methods, and also noted that the random diffusivity could potentially decrease the simulation accuracy noticeably.

Beletsky et al. [2007], who developed the TRACE algorithm, considered the transport of the yellow perch larvae as well as a biological model. The biological model is not similar to that of a virus as the larvae are significantly larger in size and weigh significantly more. These are both factors considered in the individual-based biological model, whilst only the transport model is

[1]The response of the ocean surface to an impulse, such as wind, can be modelled with non-advective and non-diffusive equations of motion. The solution to these equations of motion are used to validate the numerical model.

[2]A steady-state solution to the stream function that is obtained by integrating the Stommel equation.

considered for the virus transport model by Beletsky et al. [2007]. To develop the transport model, Beletsky et al. [2007] made use of Lagrangian equations of motion in three directions, thus tracking the particles in three dimensions. The study assumed a **bilinear variation in the horizontal currents. A linear interpolation from the edges of the cell in which the particle is located is used to calculate the derivatives of the velocity with respect to the x and y directions, which is assumed to lie in the horizontal plane.

Blanke and Raynaud [1997] developed the ARIANE algorithm, which computes the three-dimensional trajectory of a particle. The study made use of the Aeakawa C-grid, which positions the grid such that the velocities in the horizontal plane, u and v are known in the centre of the edges of the grid cells. The algorithm in Blanke and Raynaud [1997] assumed a stationary velocity field in time, this means that the velocity field is constant throughout time and space. The Blanke and Raynaud [1997] algorithm was derived from the assumption that the divergence of the velocity field is zero, and thus there are no sinks (destruction of fluid) or sources (creation of fluid) within the infinitesimal cell. It is also assumed that the transport, which is a scaled velocity field, is linearly dependent on the direction in which it travels between the faces. A limitation of the Blanke and Raynaud [1997] algorithm is that the position is only calculated on the edges of the cell. A strength of the algorithm is that it can be used to compute the origin of the particle if the current position is known, by making use of backtracking methods. The backtracking is done by backwards integration over the particle's path. The algorithm also considered the dimensional non-divergence of the flow. Fabbroni [2009] found this algorithm to display counter-intuitive behaviour as the algorithms were found to be more accurate as the timestep increased, a non-linear correlation was found between the Lagrangian timestep and the absolute error.

Toral [2014] discussed the random diffusion of a particle as Brownian motion, which was derived from a random walk model. Toral [2014] gave a deeper understanding on the derivation of the random diffusion experienced by a particle that is the size of a virus particle suspended in a fluid.

Cage dynamics

Once both the particle tracking and population models are considered, the effect that the cage has on the velocity field is important, as the cage acts as a porous obstruction within the velocity field. This can be modelled with the use of Computational Fluid Dynamics (CFD). When modelling the problem with the use of CFD, the inclusion of turbulence models should also be considered.

Bi and Xu [2018] considered the effect of bio-fouling on fish cages and the influence it has on the velocity field, along with the influence of the height of the cages. Bi and Xu [2018] also considered various arrangements of cages, and

commented on the optimal farm orientation within a current. The simulations were done in ANSYS FLUENT and was based on the Navier-Stokes equations. A cylindrical porous ring was used to simulate the cage, with a decreasing porosity as the bio-fouling increased. Bi and Xu [2018] gave insight into the modelling of cages and their effect on the velocity field. Bi and Xu [2018] found that velocities dropped to around half the incoming velocity within the farms, and around 80% of the incoming velocity after the cage. Bi and Xu [2018] showed that the cage has a significant influence on the velocity field and subsequently on the transmission of diseases between farm cages.

A limitation of Bi and Xu [2018] is that the fish seem to not have been included in the model, which in practice should have been included. Bi and Xu [2018] considered a realisable $k - \epsilon$ turbulence model within the domain.

Winthereig-Rasmussen et al. [2016] also modelled the fish cages as porous obstacles to the flow. In Winthereig-Rasmussen et al. [2016] the simulation was conducted in ANSYS FLUENT and the Reynolds average Navier-Stokes (RANS) and large eddy simulations (LES) equations were used to describe the system. Similarly to Bi and Xu [2018], Winthereig-Rasmussen et al. [2016] included a realisable $k - \epsilon$ turbulence model within the domain. Winthereig-Rasmussen et al. [2016] uses experimental data from salmon farms at Gulin in the Faroe Islands to compare to the computational model, and found that the flow was over-predicted by 50% through the cages in comparison to the physical farm. Winthereig-Rasmussen et al. [2016] found that the diameter of the cage has a negligible influence on how the velocity is influenced. They also considered various net solidities (the solidity is the complement of porosity) and found a small deviation of the results, thus finding that the cage deformation had a negligible effect on the velocity field.

Chen and Christensen [2016] made use of volume averaged Reynolds averaged Navier-Stokes (VARANS) equations as the governing equations on an unstructured (or irregular) grid. They aimed to derive new expressions to describe resistance coefficients in porous media models. The study by Chen and Christensen [2016] found after validating the numeric model, the porous media model was feasible.

Patursson [2008] simulates the system in ANSYS FLUENT while using Reynolds averaged Navier-Stokes (RANS) and large eddy simulations (LES) as the governing equations. Patursson [2008] thoroughly investigated the resistance coefficients of the porous media that represents a net panel.

Turner et al. [2015] experimentally determined the wake around two fish cages in succession. They found that the flow through the centre of the cage was significantly slower than computational studies by Bi and Xu [2018] and Winthereig-Rasmussen et al. [2016] have suggested. The experimental results also confirmed the presence of vortex shedding around the cages, as discussed in Chapter 3.

Alternative particle tracking software

There are currently various software and post-processing solvers to track particles in a fluid. However, the particles tracked are subjected to inertial force and a buoyant force. The particles tracked in this study, as will be discussed in Section 3.2, are considered to have a negligible size and weight, and will be tracked as fluid elements. For this purpose, a **robust particle tracking model is not necessary and an in-house post-processing solver was written to simulate the motion of the particles within the current. Some alternative software that could be used to track particles include OpenFOAM, OpenDrift and MODPATH.

OpenDrift

OpenDrift is an open-source Python package that allows users to make use of a time series of velocity and wind data in a NetCDF file to track the likely path that a particle submerged in the fluid would take [Dagestad et al., 2018]. This package contains multiple modules, OpenDrift, OpenOil, Leeway, PelagicEgg and OpenBerg. These modules can be used for tracking the trajectory of oil spills, rigid objects, plankton, fish eggs, larvae, ice bergs, etc. The package was developed by Knut-Frode Dagestad and his team at the Norwegian Meteorological Institute Norway.

The package was not used as part of this study because the software package requires NetCDF files and geographical data to simulate the spread at a specific location in the ocean. Due to the scale of the fish cages, the information needed would have to be high resolution. OpenDrift may be used if the scale is appropriate, and should be considered if the spread of disease between separate fish farms is to be calculated.

OpenFOAM

OpenFOAM is an open-source C++ toolbox used to simulate fluid dynamics systems. OpenFOAM version 1912 has a module `uncoupledKinematicParcelFoam` that is a transient solver for passive transport of a particle. This solver makes use of flow fields that have already been computed in OpenFOAM and allows for particle clouds to be tracked within the flow field, for example, solid particles within an incompressible flow. Although there are other solvers that can be used in OpenFOAM to track particles, `uncoupledKinematicParcelFoam` will be considered. The solver `uncoupledKinematicParcelFoam` takes phase-coupling mechanisms and particle-particle interactions into consideration. The particle tracker is also valid for larger, heavier particles [Kasper, 2017]. It is possible to write custom OpenFOAM solvers if there are no solvers available for a specific problem, however that is beyond the scope of this study.

MODPATH by MODFLOW

MODFLOW is a U.S. Geological Survey modular finite-difference flow model, for groundwater flow [Pollock, 2016][Pollock, 2012], and is an open-source flow model written in Fortran. MODPATH is a post-processing particle tracking algorithm that works with the outputs of MODFLOW. The algorithm used by MODPATH is discussed in Section 3.3.1 as the algorithm could be used for any flow and is not limited to groundwater.

1.2 Overview of this study

In this section, the work done in this study will be discussed.

Before considering the mathematical model, the physical set-up of the cages within a farm must be considered. This study focuses on farms consisting of cylindrical cages. The cages are constructed from a floating ring, a sinker ring, that keeps the cage in place, and a net held in place by connective twine, as illustrated in Figure 1.1

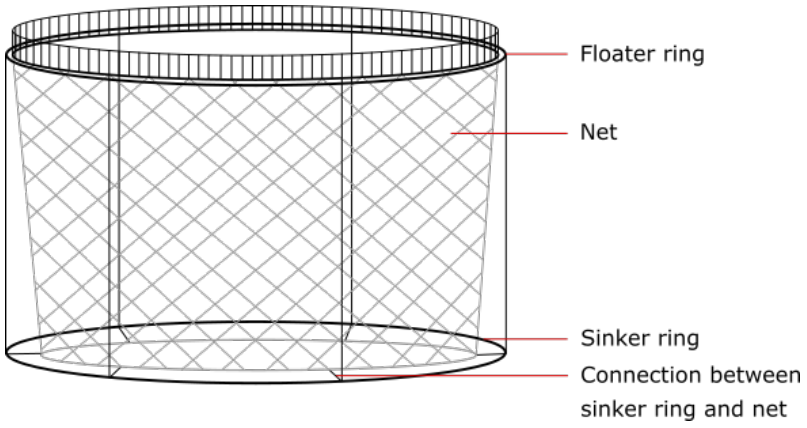


Figure 1.1: The cylindrical salmon cage within the farm that will be considered in this study.

The cages are placed in the open ocean in a grid pattern, often in two rows as can be seen in Figure 1.2a [Bi and Xu, 2018; Klebert et al., 2015; Winthereig-Rasmussen et al., 2016]. Although this is not the only manner in which a farm can be constructed, it is the set-up that will be considered in this study. It should be noted that the cages can be placed in a cluster, where multiple cages are attached on either side to a central beam structure, however, the dynamics of such systems are not considered.

(a) A salmon farm in Skye, United Kingdom, where cylindrical cages are placed in a grid pattern. Photo © John Allan (cc-by-sa/2.0)[Allen, 2001].

(b) A salmon farm in Ruakaka Bay in New Zealand where cubic cages are placed in a grid pattern with no separation distance between cages. Photo © Sid Mosdell (cc/2.0) [Mosdell, 2007].

Figure 1.2: Two separate farm set-ups with different cages.

The farms are generally placed within fjords, where a slow moving current can pass through the cages which is not strong enough to upset the floating ring, but is strong enough to remove waste from the cages.

This however opens up the possibility that infections in upstream cages can spread to cages downstream as the virus particles are suspended within the fluid and follow its path.

This study combines a population model with a Lagrangian particle tracking model to simulate the spread of virions and bacteria as they are generated within an initially infected cage.

Within the initially infected cage, the infection spread can be modelled by an epidemiological model. The epidemiological model predicts the population distribution between different states of susceptibility, exposure, infection and recovery. This is modelled in Chapter 2.

At each timestep, the infected fish shed virions, the virions are then tracked in two dimensions as a cohort. The cohort is modelled as a single particle, with an advection and diffusion term that contributes to the particles' motion. The advection term accounts for the constant motion of the current that makes up the majority of the virions' motion. The diffusion term accounts for the random motion that occurs as a result of the motion of the fluid molecules on a molecular scale.

The algorithms to track these cohorts, as well as the code that is implemented are discussed in Chapter 3.

The presence of porous obstacles, such as a cylindrical net with fish in a fjord, may distort the velocity field [Bi and Xu, 2018; Winthereig-Rasmussen et al., 2016]. The effect of the cages can therefore not be neglected during hydrodynamic modelling. The cages are modelled in two dimensions as porous rings

with a constant velocity on the inlet face and simulated with OpenFOAM with varying turbulence models. The turbulence models considered include no turbulence model, $k-\epsilon$ turbulence model and realisable $k-\epsilon$ turbulence model. The realisable $k-\epsilon$ turbulence model has an improved formulation for the turbulent viscosity, unlike the $k-\epsilon$ turbulence model, the term for turbulent viscosity is not a constant but rather a variable. The second difference between the two models is that the equation for dissipation rate is different.

OpenFOAM outputs the velocity field for each case. These velocity field outputs from OpenFOAM are then used in conjunction with the in-house particle tracking computer code to determine where the infection is likely to spread, as well as comparing the difference between the different turbulence models.

The results of these simulations are then discussed in Chapter 6.

1.3 Objectives

The focus of this study is to evaluate whether it is feasible to simulate the spread of a waterborne infection between cages within a salmon farm. The aim is to achieve this by making use of a Lagrangian particle tracking model coupled with both an epidemiological model and a computational model for the velocity field of the cages within the open ocean.

The objective of the disease model is to determine the number of virions that are shed at each timestep. This model will inform the coupled particle tracking model. This study also aims to compare the disease models considered in the literature. This is done by means of a shedding model that accounts for the number of particles created at each timestep and are added to the particle tracking model.

The particle tracking model aims to simulate the spread of disease particles within a predefined domain. This study also aims to compare the particle tracking models considered in the literature. The objective is to simulate biological particles that are shed within a confined area. The number of particles that need to be tracked will lose effectiveness over time and require a decay model.

The coupled disease and particle tracking models will be used as post processing on a velocity field generated with a CFD model. An objective of the OpenFOAM CFD model is to evaluate the effect that the porous ring has on the surrounding velocity field and the effect it has on the dispersion of the virus particles, with or without the inclusion of turbulence models. A secondary aim of the present study is to assess the various turbulence models by means of comparison between the $k-\epsilon$, realisable $k-\epsilon$ and no turbulence model.

Chapter 2

Disease models

There are many considerations in the construction of the population model for fish. The first consideration is the time scale of the infection outbreak, and whether the population is in equilibrium or whether birth and natural mortality rates are considered. The second consideration is whether the deceased individuals form part of the recovered population. The third consideration is whether the pathogen is likely to kill the host, which ties in with the final consideration, being to consider the recovered individual, i.e. is the individual immune to reinfection or susceptible once it has recovered.

There are different models that take a different approach to each consideration and are thus used for different viruses, they are the:

- Susceptible-Infected (SI) model, [Allen, 1994]
- Susceptible-Infected-Recovered (SIR) model, [Allen, 1994]
- Susceptible-Infected-Susceptible (SIS) model, [Allen, 1994]
- Susceptible-Infected-Virus (SIV) model, [Milliken, 2017; Milliken and Pilyugin, 2016]
- Susceptible-Exposed-Infected-Recovered (SEIR) model, [Institute for Disease Modeling, 2019; Salama and Murray, 2013,0]
- Susceptible-Exposed-Infected-Recovered-Susceptible (SEIRS) model [Institute for Disease Modeling, 2019].

These models can be used with vital dynamics (demography) or without, which in each case changes the various models as a term for birth rate is added to the susceptible population and death rate terms are added to all of the populations. The vital dynamics should be used if the life span of the epidemic is long enough that vital dynamics can occur.

The model that is chosen for the population depends on the type of pathogen, as mentioned above. The three fish diseases considered in Salama and Murray [2013] were

- Infectious pancreatic necrosis virus type (IPNV-type),
- Infectious salmon anaemia virus type (ISAV-type),
- Aeromonas salmonicida bacteria type (AS-type).

The two models considered for investigation are SIR and SEIR to represent the pathogen spread through the population. The following chapter considers the behaviour of the population models, but not yet in the context of the viruses discussed in this study.

2.1 Susceptible infected recovered (SIR)

Initially a discrete SIR model with no vital dynamics was considered [Allen, 1994].

The susceptible population is represented by,

$$S_{k+1} = S_k - \beta \frac{S_k I_k}{N} \Delta t, \tag{2.1}$$

the infected population is represented by,

$$I_{k+1} = I_k - \gamma I_k \Delta t + \beta \frac{S_k I_k}{N} \Delta t, \tag{2.2}$$

and the recovered population is represented by,

$$R_{k+1} = R_k + \gamma I_k \Delta t, \tag{2.3}$$

where, β is the contact rate, or the average number of contacts needed to pass the infection to an individual in the susceptible population. The coefficient γ is the relative removal rate which gives the probability that an infected individual will be removed from the infected population. N is a normalisation parameter, or the total number of fish in the population. The timestep between the discrete time intervals is denoted by Δt. The terms S_k, I_k, and R_k are the number of individuals in the susceptible, infected and recovered categories, respectively, in the kth timestep.

The term $\beta \frac{S_k I_k}{N} \Delta t$ in equations (2.1) and (2.2) is the number of new infected individuals in each timestep, where $\beta S_k I_k$ is the number of contacts between

infected individuals and individuals that can become infected multiplied by the contact rate that ensures that the infection will transfer during a contact in the timestep.

In equations (2.2) and (2.3) the term $\gamma I_k \Delta t$ is the number of individuals that recover from the infection in each timestep.

The subscript used in the literature represents the time $k\Delta t$, such that S_k is the susceptible population at the time $t = k\Delta t$.

Figure 2.1 is an illustration of equations (2.1) to (2.3) and the progression of the infection within the initially infected farm. The susceptible population becomes infected by $\beta \frac{S_k I_k}{N}$ fish per timestep and the infected population then recovers by γI_k fish per timestep.

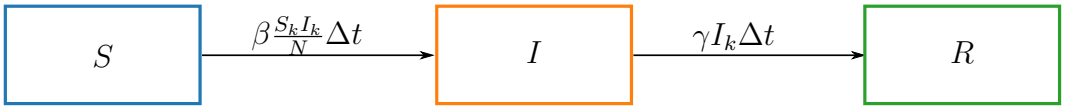


Figure 2.1: A flow diagram describing the population distribution with an SIR model without vital dynamics.

The SIR model without vital dynamics is modelled by equations (2.1) to (2.3) and are represented in Figure 2.2 with the time-difference equations. The susceptible curve shows that the majority of the population are initially susceptible to infection. As the population becomes infected and the infected curve begins to increase, the susceptible population decreases. The infected population, as shown in the curve in Figure 2.2, initially increases gradually and after approximately 10 days, the infected population begins to increase exponentially, once half the population is infected, the transmission rate decreases, and the infected population begins to decrease as the fish continue to recover. As this happens, the recovered population continues to increase until the entire population is recovered.

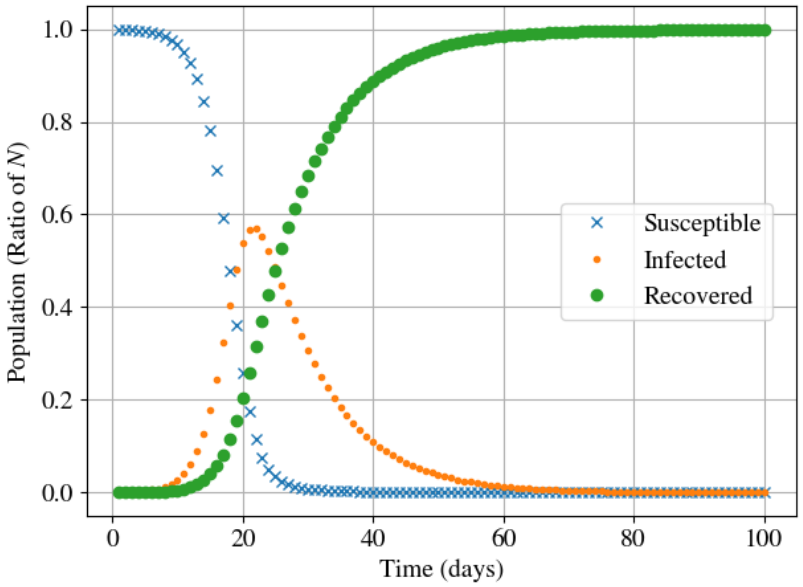


Figure 2.2: The population curves of the SIR model without vital dynamics, modelled with difference equations. The values used are not representative of the disease but instead are for illustrative purposes, $\beta = 0.6$ and $\gamma = 0.1$.

Considering the SIR model with vital dynamics, where, μ and ν are the birth and death rates of the population, respectively. The difference equations change as follows [Institute for Disease Modeling, 2019].

$$S_{k+1} = S_k + \mu N \Delta t - \nu S_k \Delta t - \beta \frac{S_k I_k}{N} \Delta t, \tag{2.4}$$

$$I_{k+1} = I_k - \nu I_k \Delta t - \gamma I_k \Delta t + \beta \frac{S_k I_k}{N} \Delta t, \tag{2.5}$$

$$R_{k+1} = R_k - \nu R_k \Delta t + \gamma I_k \Delta t. \tag{2.6}$$

The terms $\nu S_k \Delta t$, $\nu I_k \Delta t$, and $\nu R_k \Delta t$ are the number of individuals that die in the susceptible, infected and recovered categories, respectively, in each timestep. The term $\mu N \Delta t$ is the number of births in each timestep.

In Figure 2.3, the total time is twice that of Figure 2.2 to illustrate the decrease in the recovered population as the fish die of natural causes, and there is an increase in susceptible population as new fish are born. The general trend of the three susceptible, infected and recovered curves display similar behaviour to those in Figure 2.2 and the infection is removed from the system before 100 days.

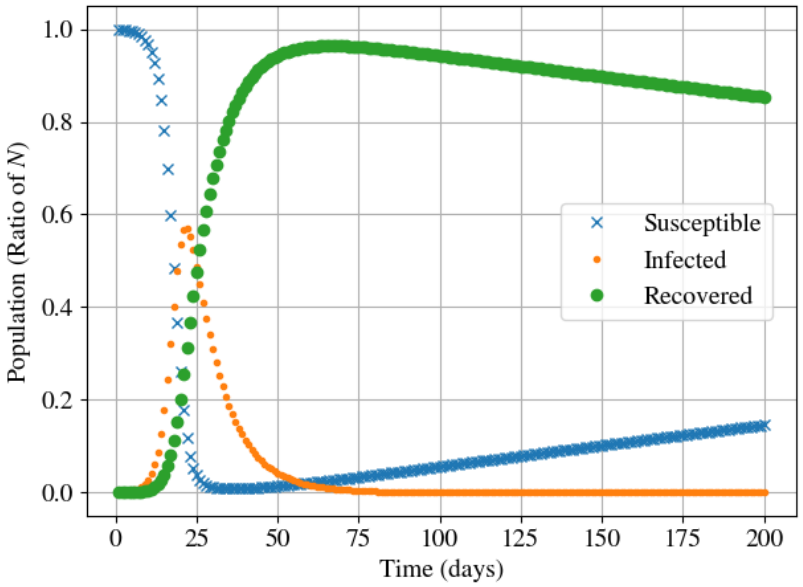


Figure 2.3: The population curves of the SIR model with vital dynamics, modelled with the discrete time-difference equations, where the birth and death rates, $\mu = \nu = 0.001$. The values used are not representative of the disease but instead are for illustrative purposes, $\beta = 0.6$ and $\gamma = 0.1$.

If the birth and death rates are increased by an order of magnitude from that in Figure 2.3, i.e. $\mu = \nu = 0.01$, the infection is shown to persist as is illustrated by Figure 2.4. This is, however, assuming that the infection affects both adults and newly born individuals in the same manner.

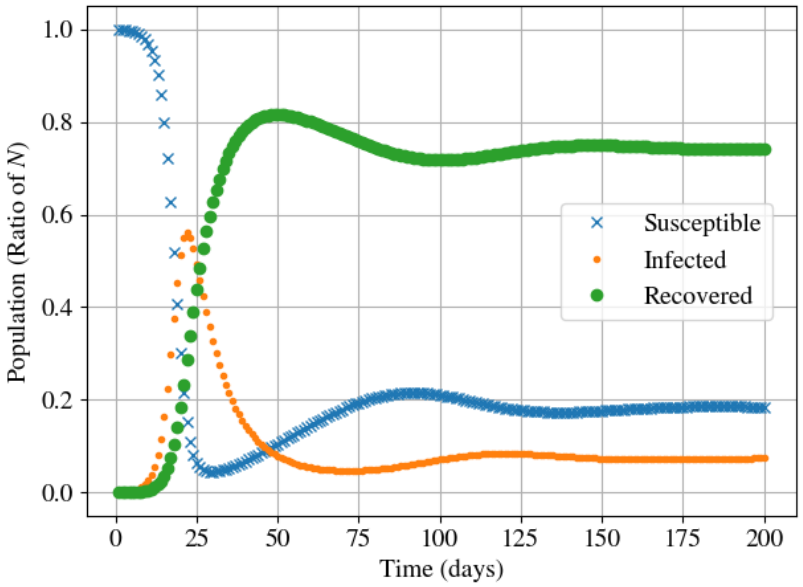


Figure 2.4: The population curves of the SIR model with vital dynamics, modelled with coupled differential equations, where the birth and death rates, $\mu = \nu = 0.01$. The values used are not representative of the disease but instead are for illustrative purposes, $\beta = 0.6$ and $\gamma = 0.1$.

2.2 Susceptible exposed infected recovered (SEIR)

The SEIR model works well for diseases that have a period in which the individual is carrying the disease and is infected but cannot yet infect others. These individuals then fall within the exposed population. The SEIR model is somewhat similar to the SIR model. The discrete time equations without vital dynamics are as follows [Salama and Murray, 2013], where the susceptible population is represented by,

$$S_{k+1} = S_k - \beta \frac{S_k I_k}{N} \Delta t, \tag{2.7}$$

the additional exposed population is then represented with the transmission rate β and the infection rate σ,

$$E_{k+1} = E_k(1 - \sigma \Delta t) + \beta \frac{S_k I_k}{N} \Delta t. \tag{2.8}$$

The infected population is represented by the infection rate and the removal rate γ,

$$I_{k+1} = I_k(1 - \gamma \Delta t) + \sigma E_k \Delta t, \tag{2.9}$$

and finally, the recovered population is represented by,

$$R_{k+1} = R_k + \gamma I_k \Delta t. \tag{2.10}$$

The terms in each equation can be broken down as follows, S_k, E_k, I_k and R_k, are the terms that refer to the size of the population in the susceptible, exposed, infected and recovered populations respectively. The total number of fish is again denoted by N.

In equations (2.7) and (2.8) the term $\beta \frac{S_k I_k}{N}$ is consistent with the term in Section 2.1. The term σE_k is the number of exposed individuals that then become infected. The term γI_k refers to the number of infected individuals that recover from the infection, or are immune or isolated.

Figure 2.5 illustrates equations (2.7) to (2.10) and the manner in which the population distribution changes. Unlike the SIR model in Figure 2.1, there is now the addition of the exposed population. At each timestep there are $\beta \frac{S_k I_k}{N}$ individuals that were susceptible to the virus and are now exposed and at risk of infection, and $\sigma E_k \Delta t$ individuals that were exposed and are now infected.

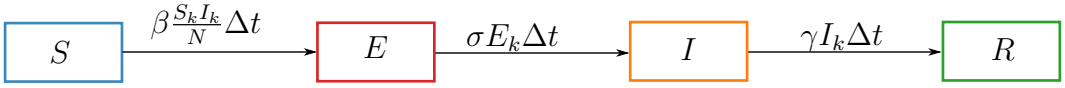


Figure 2.5: The flow diagram of the population distribution of a SEIR model with no vital dynamics.

Salama and Murray [2013] did not discuss the handling of individuals that have died, although it mentions that the time scale of the virus is much smaller than the population growth, the assumption is that the number of individuals that died is equal to the number of individuals that were born.

Figure 2.6 illustrates the behaviour of the discrete equations (2.7) to (2.10), and the accompanying sketch in Figure 2.5. In Figure 2.6 the behaviour of the susceptible, infected and recovered curves are similar to those in Figure 2.1. The addition of the exposed curve illustrates the number of individuals at risk of infection at any given time. This peaks at around 45 days and is under 25% of the population, although this is dependent on the infection parameters.

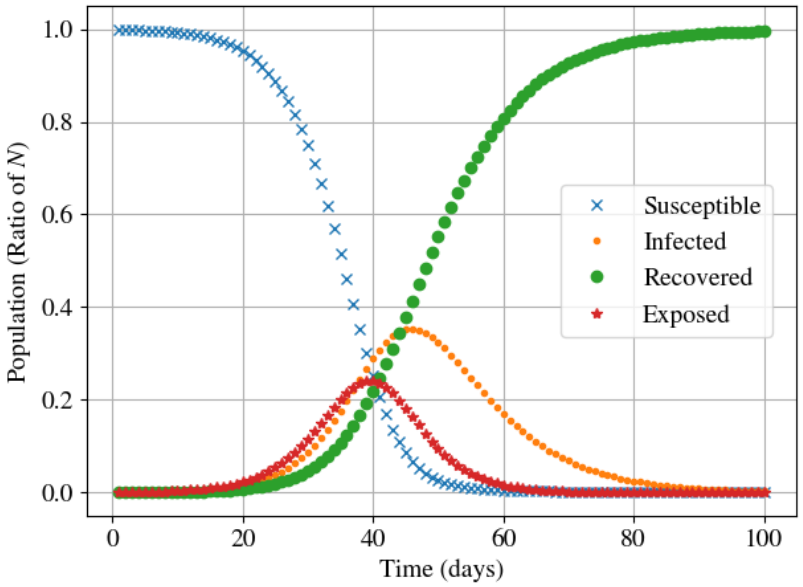


Figure 2.6: The population curves of the SEIR model without vital dynamics, modelled with time difference equations. The values used are not representative of the disease but instead are for illustrative purposes, $\beta = 0.6$, $\gamma = 0.1$ and $\sigma = 0.2$.

If vital dynamics are then added to the system, similar difference equations are obtained to those in the SIR model case [Institute for Disease Modeling, 2019], namely,

$$S_{k+1} = S_k + \mu S_k \Delta t - \nu S_k \Delta t - \beta \frac{S_k I_k}{N} \Delta t, \tag{2.11}$$

where the additional exposed population is then represented with the transmission rate β and the infection rate σ,

$$E_{k+1} = E_k(1 - \sigma \Delta t) - \nu E_k \Delta t + \beta \frac{S_k I_k}{N} \Delta t. \tag{2.12}$$

The infected population is represented by the infection rate and the removal rate γ,

$$I_{k+1} = I_k(1 - \gamma \Delta t) - \nu I_k \Delta t + \sigma E_k \Delta t, \tag{2.13}$$

and finally, the recovered population is represented by,

$$R_{k+1} = R_k - \nu R_k \Delta t + \gamma I_k \Delta t. \tag{2.14}$$

The terms in the time different equations with vital dynamics are similar to those in the SIR model with vital dynamics. When vital dynamics are added,

it is illustrated in Figure 2.7 that the maximum number of recovered does not reach the maximum population, as more fish are born susceptible, and the pathogen in this case does not die out, thus it can be seen that if the birth rate is high enough, the infection persists.

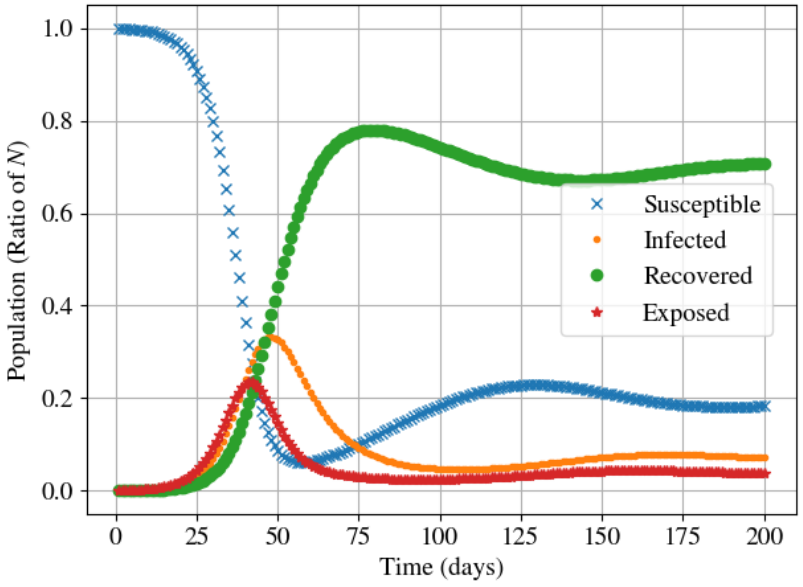


Figure 2.7: The population curves of the SEIR model, modelled with discrete time-difference and vital dynamics over a longer time period, where the birth rate, $\mu = 0.01$. The values used are not representative of the disease but instead are for illustrative purposes, $\beta = 0.6$, $\gamma = 0.1$ and $\sigma = 0.2$.

If the birth rate is decreased by an order of magnitude as in Figure 2.8, this is not the case, and the infection dies out. Although, with vital dynamics, the susceptible population will never tend to zero, no matter what the starting parameters are, as new fish are born susceptible in this case.

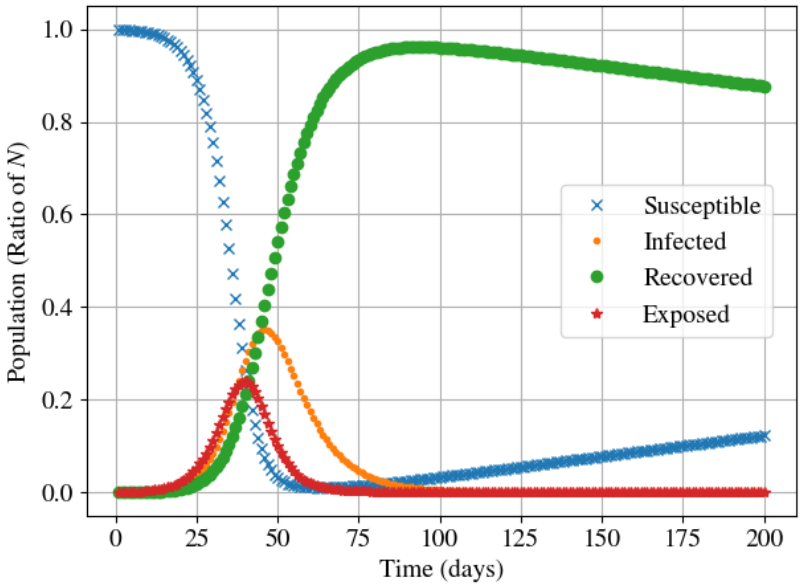


Figure 2.8: The population curves of the SEIR model, modelled with discrete time-difference equations and vital dynamics, where the birth rate, $\mu = 0.001$. The values used are not representative of the disease but instead are for illustrative purposes, $\beta = 0.6$, $\gamma = 0.1$ and $\sigma = 0.2$.

In this study, salmon farming is considered, and it seems reasonable to assume that all salmon in one farm are of the same or similar ages considering that they are hatched and transferred en masse to a cage. It is also reasonable to assume that no salmon are added to the cage during an infection outbreak, and thus the model chosen will not consider vital dynamics.

2.3 Alternative disease models

Milliken and Pilyugin [2016] conducted a study that considered the virus transfer between farmed fish populations and wild fish populations. The farm was considered as one patch, and the wild fish was the second patch. This two-patch deterministic model had both a host-to-host transmission component as well as an environmental transmission component of the infection. The interaction between the wild and farmed fish populations was through the diffusion of the virus in the ocean. The model also considered the regular vital dynamics of both fish populations, however, the infected fish cannot reproduce but experience the regular mortality as well as the disease related mortality. Generally, a susceptible, infected, virus (SIV) model does not look at populations of individuals but rather susceptible cells, infected cells and virus cells. Milliken and Pilyugin [2016], however, made use of the susceptible and infected populations and number of virus cells. The model used by Milliken and Pilyugin [2016]

also made use of shedding and clearance of the virions, similar to the shedding and decay in the study by Salama and Murray [2013] and Salama and Murray [2011].

Milliken [2017] showed that a Markov chain is a better approximation of the infection spread over multiple areas. The model presented by Milliken and Pilyugin [2016] and Milliken [2017] was not considered in this study.

2.4 Choosing the percentage of fish exposed

It is important that the initial number of exposed individuals, E_0, and infected individuals, I_0, in a population is realistic. The effect the choice of initial values has on the spread of disease will be discussed in this section.

Earlier in Chapter 2, the population models made use of $E_0 = 1$, $I_0 = 1$ for the respective initial exposed and infected populations to display the behaviour of the population model. If these values for the initial populations are used along with the disease parameters for a realistic cage with 9×10^4 fish [Watershed Watch Salmon Society, 2004], the disease does not spread throughout the population and the initial cage cannot be considered infected, as can be seen in Figure 2.9, as the entire population remains susceptible throughout the timespan. This happens because a single fish that is exposed to another single fish, and no other fish, will not likely cause a spread of infection throughout the population of 9×10^4 fish. The single infected fish only being exposed to a single fish is not a realistic situation as it is unlikely that in a fish cage, where the fish live in such close proximity, only one fish would be exposed to the single infected fish. It is more likely that the single infected fish will come in contact with a portion of the total number of fish. This raises the question of how to make an appropriate choice for the initial exposed population, such that a realistic infection is observed.

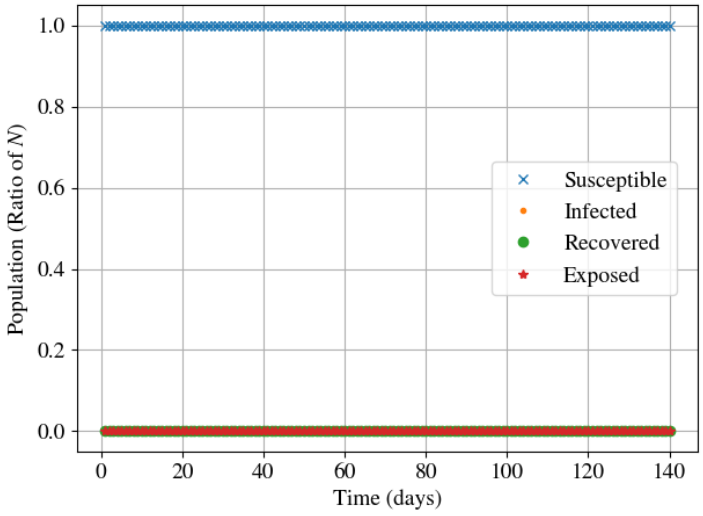


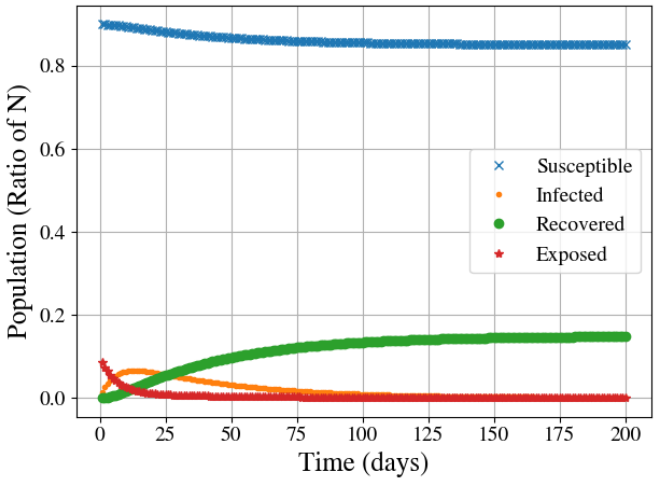
Figure 2.9: The population curves of the SEIR model with the parameters for ISAV-type, where the initial infected and exposed populations are the minimum of one fish per population.

The initial exposed population can be chosen as a percentage of the total population, as in Figures 2.10a, 2.10b, 2.10c, where the percentages are 10%, 50% and 99% respectively. It is illustrated in these figures, that the severity of the virus outbreak is proportional to the percentage of the population that is initially exposed to the infection.

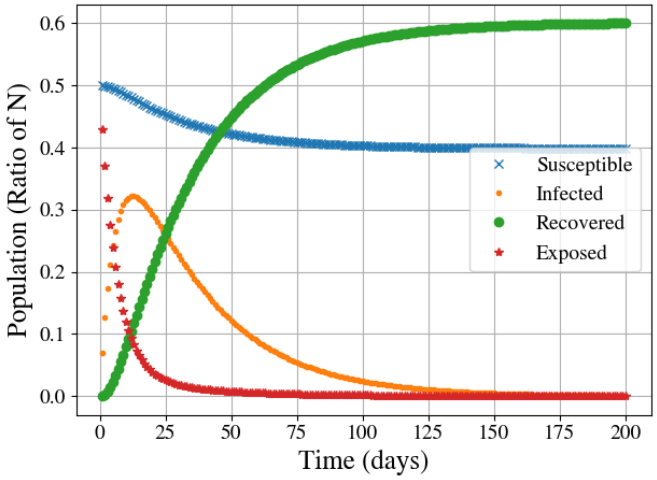
If 10% of the population is initially exposed, the maximum number of individuals infected at a given time is approximately 5% and the total number infected, as indicated by the number of recovered individuals after the infection has cleared, is approximately 13% of the population.

If the initial exposed population is 50% of the total population, the maximum infected population at a given time is approximately 30% and the total recovered individuals is approximately 57%

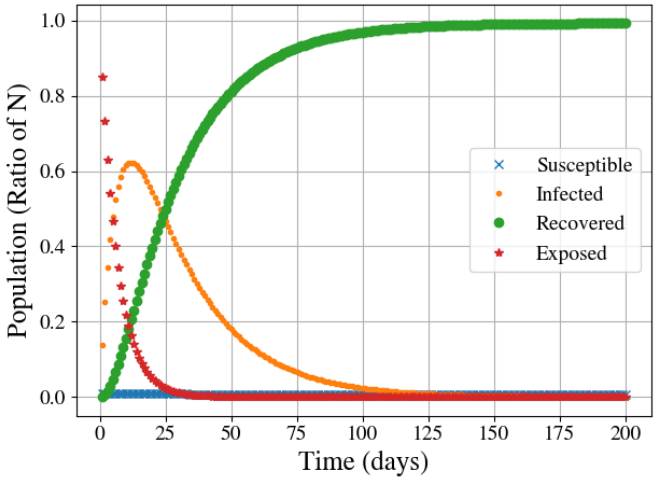
However, if 99% of the population is exposed, it may be better to use the SIR model instead of the SEIR model, as almost every fish is exposed to the virus, and thus for practical purposes, all susceptible fish are also exposed. The maximum infected at a given time is approximately 61% and the recovered is approximately 94% and thus effectively the entire population becomes infected with the virus.



(a) The population curves when the initial exposed population is chosen as 10% of the total population, i.e. $E_0 = 0.1N$.



(b) The population curves when the initial exposed population is chosen as half of the total population, i.e. $E_0 = 0.5N$.



(c) The population curves when the initial exposed population is chosen as 99% of the total population, i.e. $E_0 = 0.99N$.

Figure 2.10: The initially exposed population E_0 set to various percentages of the total population.

Throughout this chapter the various possible population models used to track the spread of infection within the cages are discussed. The SEIR model is chosen as the viral infection considered has an incubation period in which fish are exposed to the virus but do not yet show signs of infection. The initial conditions of the infection are also chosen as a single infected individual, $I_0 = 1$ and a conservative 80% of the population within the cage are initially exposed to the virus, $E_0 = 0.8N$ to ensure that a significant portion of the population is indeed infected.

In the following chapter the particle tracking algorithm is investigated. This particle tracking algorithm will be used in conjunction with the disease model to ensure that the number of particles tracked are representative of the physical system.

Chapter 3

Particle tracking

In this chapter, the particle tracking algorithm and all of the components, such as the most effective clustering, properties of the virions, particle tracking algorithms, diffusion, shedding, decay and the minimum infection threshold, are discussed. All of these components are combined to construct the final algorithm used to simulate the spread of a viral waterborne disease.

3.1 Choosing cluster sizes

When modelling the transport of individual particles, it is often not computationally efficient to track each individual virion. The efficiency of clustering virions into various cohort sizes is determined by clustering 4.904×10^{15} test particles and comparing the percentages of the particle clusters that either reached or did not reach the second cage.

The percentage of the particle clusters reached is then compared between various cluster sizes and a cluster size is chosen for the future simulations.

Table 3.1 gives the percentage of various size cohorts that reached the second cage.

Table 3.1: Tabulated results illustrating the distribution of passive virions that either reach the second cage, or do not, within a 200 min time frame. The 200 min time frame was chosen at random.

Cohort size			
	Cluster size: 1×10^{12}	Cluster size: 2×10^{12}	Cluster size: 2×10^{13}
Run	% Clusters Reached	% Clusters Reached	% Clusters Reached
Run 1	96.6	95.9	97.7
Run 2	96.2	96.3	96.0
Run 3	96.5	96.8	94.8
Run 4	96.4	95.7	94.4
Run 5	95.9	96.0	94.2
Run 6	96.0	96.3	97.7
Run 7	96.5	96.3	97.7
Average	96.3	96.2	96.1

From the results in Table 3.1, it appears that the size of the cohort of virions does not significantly influence the number of cohorts that reach the cage, between the cluster sizes chosen, which allows the cohort size to be slightly larger to minimise computational cost. It was found in practice, however, that using a cluster size that is too large can result in a sparse concentration map. Through trial and error it was found that a good cluster size was 5×10^{12} virions per cluster.

3.2 Particle properties

The scale of the particles being tracked is an important factor to consider because the forces that affect the particle's motion is dependent on the scale of the system.

The infectious pancreatic necrosis virus type (IPNV-type) has a diameter of 70 nm [Kar, 2016; Kibenge and Kibenge, 2016] the infectious salmon anaemia virus type (ISAV-type) is much larger with almost twice the diameter of 90 nm to 140 nm [Jamieson, 2007; ViralZone, 2015] and length of 13 to 15 nm [Kibenge and Kibenge, 2016].

The densities of the virions should be calculated to determine whether the non-motile rods will be suspended in the fluid, float on the surface, or settle to the ocean bed. The information on the dimensions of the virions is inconclusive, and the density of the virion could not be calculated. The weight range of virions is also too large to determine a workable density.

It is fairly safe to assume that a waterborne virus will be suspended within the fluid. As the virions are on the nano scale, the assumption of negligible inertia is realistic.

The Aeromonas salmonicida type (AS-type) bacterium is significantly larger than both virions. The AS-type is a rod-shaped, non-motile bacterium with the length of 1.3 μm to 2.0 μm and the diameter of 0.8 μm to 1.3 μm. As the bacterium is both larger and heavier than the average virion (around 18 times larger than the smallest virus particle). The bacterium will not necessarily follow the same path as a virion would. The bacterium is not considered in this study.

3.3 Mathematical models of particle tracking algorithms

In computational fluid dynamics (CFD), the Lagrangian particle tracking is a numerical technique for tracking Lagrangian particles within an Eulerian phase. It is also commonly referred to as a discrete particle simulation.

The particle tracking algorithm chosen for this study, which consists of advective and diffusive terms, will be used in conjunction with the population model, the virus decay rates and cage dynamics, to simulate the path that the particles will follow. The mathematical theory behind each particle tracking algorithm is discussed in this chapter.

An additional algorithm, the ARIANE algorithm developed by Blanke and Raynaud [1997], is discussed in Appendix C, but is not tested in this study.

3.3.1 MODFLOW particle tracking formulation

The MODPATH (Particle tracking for MODFLOW, a finite-difference solver) user manual describes the method to track the particles in a steady state flow and later look at how to generalise to transient flow [Pollock, 2016][Pollock, 2012]. Both Pollock [2012] and Pollock [2016] also considered switching between forward and backward tracking but do consider what cases would be best suited to each. The derivation by Pollock [2012] is then considered and starts with a conservation of mass equation for an infinitesimally small volume,

$$\frac{\partial}{\partial x}(\phi v_x) + \frac{\partial}{\partial y}(\phi v_y) + \frac{\partial}{\partial z}(\phi v_z) = \dot{\mathcal{V}}, \tag{3.1}$$

$$\nabla \cdot (\phi \mathbf{v}) = \dot{\mathcal{V}}, \tag{3.2}$$

where $\mathbf{v}$ is the velocity vector with the elements v_x, v_y and v_z, ϕ is the porosity, $\dot{\mathcal{V}}$ is the volume rate of water created or consumed by internal sinks or sources. It should be noted that the derivations in Pollock [2016] take the porosity of the domain into consideration, however, for the purpose of this study, a non porous domain is considered, and thus $\phi = 1$.

The three-dimensional infinitesimal fluid element is illustrated in Figure 3.1. The faces are then defined as $x_1,\ x_2,\ y_1,\ y_2,\ z_1,\ z_2$, where face x_1 is the face perpendicular to the x direction at $x = x_1$ and x_2 is in the negative direction, and similarly for the other faces.

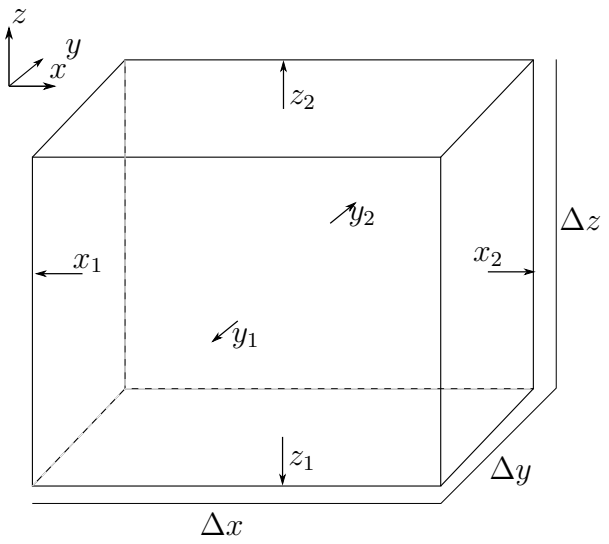


Figure 3.1: A representation of the infinitesimal fluid element cell on which the particle tracking algorithm is based.

Using the definition of volumetric flow rate Q, where $Q = vA$, v is the average linear velocity over the face of the cell and A is the area of the face over which the volumetric flow rate is being calculated. The scalar form of the velocity at each face may be written as follows,

$$v_{x_1} = \frac{Q_{x_1}}{\Delta y \Delta z}, \qquad v_{x_2} = \frac{Q_{x_2}}{\Delta y \Delta z}, \tag{3.3}$$

$$v_{y_1} = \frac{Q_{y_1}}{\Delta x \Delta z}, \qquad v_{y_2} = \frac{Q_{y_2}}{\Delta x \Delta z}, \tag{3.4}$$

$$v_{z_1} = \frac{Q_{z_1}}{\Delta x \Delta y}, \qquad v_{z_2} = \frac{Q_{z_2}}{\Delta x \Delta y}. \tag{3.5}$$

By subtracting the outflow from the inflow, the volumetric rate of production within the cell is,

$$\begin{aligned}(v_{x_1}\Delta y\Delta z + v_{y_1}\Delta x\Delta z + v_{z_1}\Delta x\Delta y)& \\ - (v_{x_2}\Delta y\Delta z + v_{y_2}\Delta x\Delta z + v_{z_2}\Delta x\Delta y) = Q_s,&\end{aligned} \tag{3.6}$$

where the left side is the net outflow of the cell per unit volume, and the right hand side is the net volumetric rate of production per unit volume from internal

sources and sinks. Dividing equation (3.6) by the volume $\Delta\mathcal{V} = \Delta x\Delta y\Delta z$ yields,

$$\frac{(v_{x_2} - v_{x_1})\Delta y\Delta z}{\Delta\mathcal{V}} + \frac{(v_{y_2} - v_{y_1})\Delta x\Delta z}{\Delta\mathcal{V}} + \frac{(v_{z_2} - v_{z_1})\Delta x\Delta y}{\Delta\mathcal{V}} = \frac{Q_s}{\Delta\mathcal{V}},$$
$$\frac{(v_{x_2} - v_{x_1})}{\Delta x} + \frac{(v_{y_2} - v_{y_1})}{\Delta y} + \frac{(v_{z_2} - v_{z_1})}{\Delta z} = \frac{Q_s}{\Delta\mathcal{V}}.$$

The velocity is then interpolated to find the velocity at point $\mathbf{p}$, $\mathbf{v}_p$ with the following equations

$$\mathbf{v}_p = \boldsymbol{\alpha}(\mathbf{p} - \mathbf{x_1}) + \mathbf{v_1}, \tag{3.7}$$

where, $\mathbf{x_1}$ is a vector with elements x_1, y_1 and z_1, and $\boldsymbol{\alpha}$ is the vector with elements α_x, α_y and α_z, which are components of the velocity gradient within the cell in the corresponding direction. The interpolation of the velocities is illustrated in Figure 3.2. The elements of the vector $\boldsymbol{\alpha}$ can be calculated as follows,

$$\alpha_i = \frac{v_{i_2} - v_{i_1}}{i_2 - i_1} = \frac{v_{i_2} - v_{i_1}}{\Delta i}, \tag{3.8}$$
$$\text{for} \quad i = x, y, z. \tag{3.9}$$

The interpolation produces a continuous velocity vector field within each cell that satisfies the equation (3.2) if the porosity is constant throughout the cell, and sinks and sources are uniformly distributed throughout the cell.

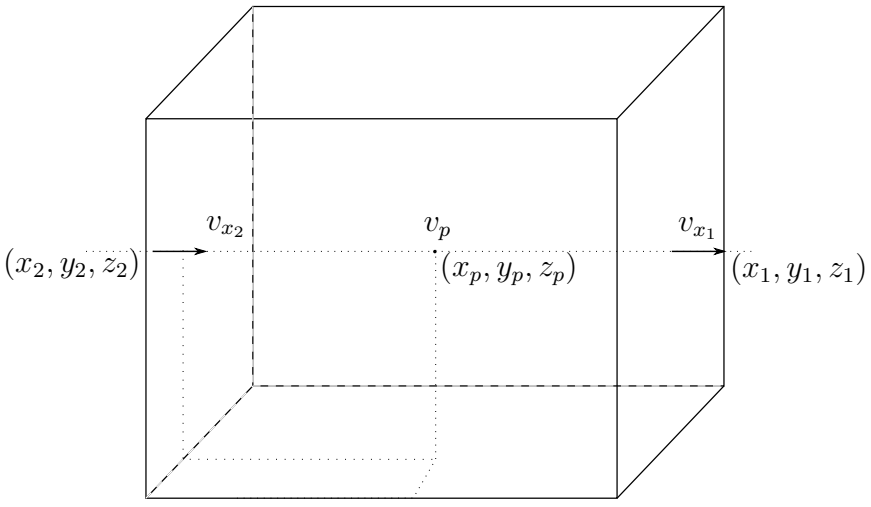


Figure 3.2: A representation of the velocity interpolation through an infinitesimal fluid element cell, to calculate the velocity at a given point p.

The particle at point $\mathbf{p}$ moves through the three-dimensional cell. The velocity in each direction is independent of the other directions [Pollock, 2016]. The

rate of change of the particle's velocity in each direction is given by,

$$\left(\frac{dv_x}{dt}\right)_p = \left(\frac{dv_x}{dx}\right)\left(\frac{dx}{dt}\right)_p. \tag{3.10}$$

Where the subscript p is used to indicate that the term is that of the particle evaluated at the particle's location $\mathbf{p} = (x_p, y_p, z_p)$. The term $\left(\frac{dx}{dt}\right)_p$ is the particle's velocity at the particle's position $\mathbf{p}$,

$$v_{x_p} = \left(\frac{dx}{dt}\right)_p. \tag{3.11}$$

From equation (3.7) it follows that,

$$\alpha_x = \frac{dv_x}{dx}. \tag{3.12}$$

Substituting in equations (3.12) and (3.11) into equation (3.10) for each direction yields,

$$\left(\frac{dv_x}{dt}\right)_p = \alpha_x v_{x_p}. \tag{3.13}$$

Similar equations are obtained in the y and z directions, respectively,

$$\left(\frac{dv_y}{dt}\right)_p = \alpha_y v_{y_p}, \tag{3.14}$$

$$\left(\frac{dv_z}{dt}\right)_p = \alpha_z v_{z_p}. \tag{3.15}$$

Equation (3.13) can be integrated between t_k and t_{k+1} (where $t_{k+1} > t_k$) and it follows that,

$$\begin{aligned} \int_{t_k}^{t_{k+1}} \frac{1}{v_{x_p}}\, dv_x &= \int_{t_k}^{t_{k+1}} \alpha_x dt, \\ \ln\left[\frac{(v_{x_p})_{t_{k+1}}}{(v_{x_p})_{t_k}}\right] &= \alpha_x \Delta t, \\ (v_{x_p})_{t_{k+1}} &= (v_{x_p})_{t_k} e^{\alpha_x \Delta t}. \end{aligned} \tag{3.16}$$

Substituting the x-component of equation (3.7) at time t_{k+1} into equation (3.16) yields,

$$\begin{aligned} \alpha_x\left((x_p)_{t_{k+1}} - x_1\right) + v_{x_1} &= (v_{x_p})_{t_k} e^{\alpha_x \Delta t}, \\ (x_p)_{t_{k+1}} &= x_1 + \frac{1}{\alpha_x}\left[(v_{x_p})_{t_k} e^{\alpha_x \Delta t} - v_{x_1}\right]. \end{aligned} \tag{3.17}$$

The formulation in equation (3.17) can be illustrated with a two-dimensional example. The starting position of a particle in cell (i, j) is known and is at position $\mathbf{p} = (x_p, y_p, z_p)$ at time t_p. For this example, it can be assumed that the velocities $v_{x_1}, v_{x_2}, v_{y_1}, v_{y_2}$ are all greater than zero, i.e. the fluid flows in across face x_1 and out across x_2 in Figure 3.1, similarly for the y direction.

The exit point of the particle from the cell can be calculated by computing the time it takes for the particle to reach each face of the cell. The process is as follows; the velocity at the starting time and position $\mathbf{p} = (x_p, y_p, z_p)$, v_{x_p} can be calculated through equation (3.7). The velocity $(v_x)_{t_{k+1}}$ can be calculated by noting that $v_x = v_{x_2}$ because $x = x_2$ at the exiting face, thus by making use of equation (3.17). The velocities in the y directions are also calculated. Once all four velocities are calculated the time it takes to exit through the face x_2 or y_2 of the cell can be calculated by rearranging equation (3.16),

$$\Delta t_x = \frac{1}{\alpha_x} \ln \left[\frac{v_{x_2}}{v_{x_p}} \right], \tag{3.18}$$

and analogously for the y direction,

$$\Delta t_y = \frac{1}{\alpha_y} \ln \left[\frac{v_{y_2}}{v_{y_p}} \right]. \tag{3.19}$$

Once these two times are calculated, there are three options for the exit face of the particle,

- $\Delta t_x < \Delta t_y$ thus the particle leaves across x_2, and enters cell $(i, j+1)$.
- $\Delta t_x > \Delta t_y$ thus the particle leaves across y_2, and enters cell $(i-1, j)$.
- $\Delta t_x = \Delta t_y$ thus the particle leaves across the corner where x_1 meets x_2, and enters cell $(i-1, j+1)$.

The smallest time taken to leave the cell is then denoted Δt_e and the point at which it exits is denoted (x_e, y_e) which can be calculated by adapting equations (3.17) for x_e,

$$x_e = x_1 + \frac{1}{\alpha_x} \left[(v_{x_p})_{t_p} e^{\alpha_x \Delta t} - v_{x_1} \right], \tag{3.20}$$

and similarly y_e,

$$y_e = y_1 + \frac{1}{\alpha_y} \left[(v_{y_p})_{t_p} e^{\alpha_y \Delta t} - v_{y_1} \right]. \tag{3.21}$$

The example in Figure 3.3, illustrates a particle's path through a cell. The times Δt_x and Δt_y are compared and it is seen that $\Delta t_x > \Delta t_y$. Thus the particle exits over face y_2.

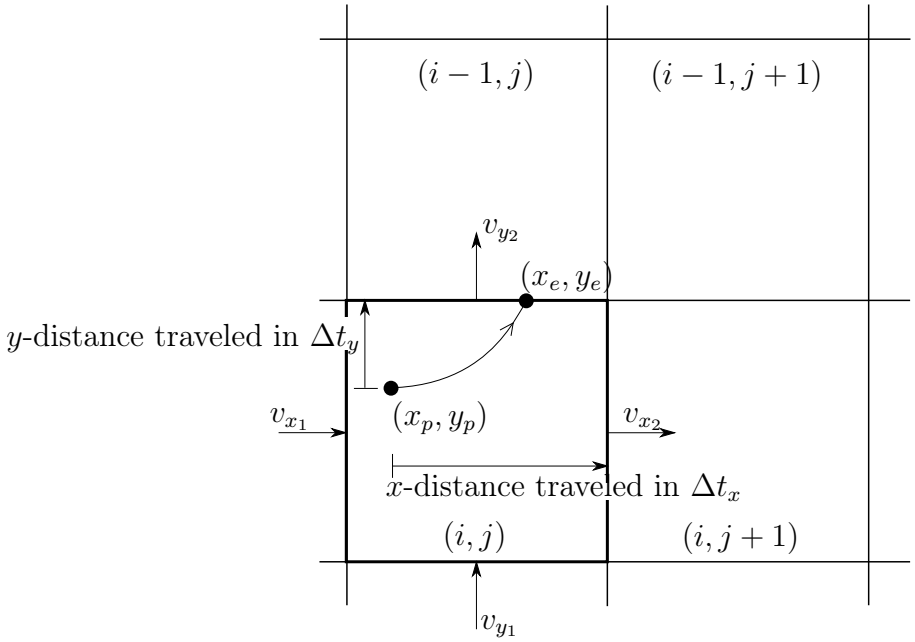


Figure 3.3: A visual representation of the position tracking of a particle through the two dimensional grid.

The calculation is repeated, cell by cell, until the particle reaches its termination point. The derivation in this section assumes that the velocity is directed such that the particle would enter over the left and bottom faces and exit over the right and top faces and the velocity changes over the cell, i.e. $\alpha_i = 0$ (where $i = x, y, z$). If any of these is true, equations (3.20) and (3.21) would evaluate to either $x_e = x_1$ or $y_e = y_1$ which would imply that the particle would be stuck on the entering face and would not exit. Consider another case where $\alpha_x = 0$ and $\alpha_y \neq 0$, but $v_x \gg v_y$; the particle would exit over the x_2 face but the derivation would suggest that $x_e = x_1$ and the particle would incorrectly exit over the y_2 face. To generalise the method further, each cell must be evaluated on where the particle exits, depending on the direction the fluid flows. The following cases in Figure 3.4 should also be considered as they do not follow the previous derivation. Figure 3.4 showcases three alternative directions that the fluid can flow (for each dimension).

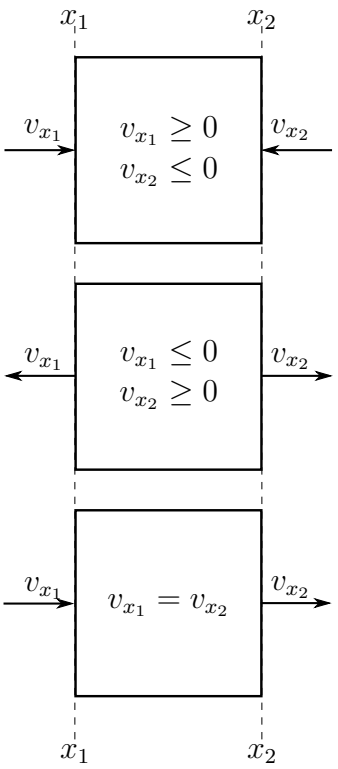


Figure 3.4: Various cases of velocity orientation on each cell boundary within the grid, resulting in the particle's overall motion within the two dimensional cell.

In the first case the particle cannot exit either x faces, and thus it must exit in another direction (depending on whether the simulation is in two or three dimensions). If it cannot exit any one of the other directional faces, then it indicates that there is a strong sink within the cell from which no particle can escape.

In the second case, the velocities in the x direction are also in opposite directions such that the particle exits out of one of the x faces, such that there exists a flow divide line in the x-direction within the cell. The direction of the particles trajectory is determined by evaluating the sign of the velocity, if the sign of the velocity is determined to be negative, the particle can potentially leave across the x_1 face. If the sign is positive, the particle has the potential to exit across the x_1 face.

When the velocities are equal and in the same direction, the velocity gradient is zero. The changes in times are then calculated with the simple expressions,

if $v_{x_1} > 0$, then,

$$\Delta t_x = \frac{x_2 - x_p}{v_{x_1}}, \tag{3.22}$$

and if $v_{x_1} < 0$, then,

$$\Delta t_x = \frac{x_1 - x_p}{v_{x_1}}. \tag{3.23}$$

For this model to be comprehensive and useful, the algorithm has to be generalised to all cases.

3.3.2 TRACE algorithm

The TRACE algorithm was derived by Beletsky et al. [2007]. This model begins with the three Lagrangian equations of motions in each direction,

$$\frac{dx}{dt} = u(x, y, z), \tag{3.24}$$
$$\frac{dy}{dt} = v(x, y, z), \tag{3.25}$$
$$\frac{dz}{dt} = w(x, y, z), \tag{3.26}$$

where (x, y, z) is the position and (u, v, w) is the velocity of the particle, and t is the time. The Taylor series expansion of the horizontal velocities about the particles horizontal positions,

$$\frac{x^{k+1} - x^k}{\Delta t} = u(x^k, y^k) + \frac{1}{2}\frac{\partial u}{\partial x}(x^{k+1} - x^k) + \frac{1}{2}\frac{\partial u}{\partial y}(y^{k+1} - y^k), \tag{3.27}$$
$$\frac{y^{k+1} - y^k}{\Delta t} = v(x^k, y^k) + \frac{1}{2}\frac{\partial v}{\partial x}(x^{k+1} - x^k) + \frac{1}{2}\frac{\partial v}{\partial y}(y^{k+1} - y^k), \tag{3.28}$$

where k is the current timestep, and Δt is the time increment.

The velocities u and v and their derivatives are bilinearly interpolated [Bennett and Hutchinson Clites, 1987]. The derivation for equations (3.27) and (3.28) can be found in Appendix B.

Equations (3.27) and (3.28) are then solved simultaneously for x^{k+1} and y^{k+1} for the particle positions at the next timestep. As the velocities are bilinearly interpolated, the expressions are formulated as follows:

$$u_p^k = m\,n\,u_{(i,j)} + (1-m)\,n\,u_{(i+1,j)} + (1-m)\,(1-n)\,u_{(i+1,j+1)} + m\,(1-n)\,u_{(i,j+1)}, \tag{3.29}$$

$$v_p^k = m\,n\,v_{(i,j)} + (1-m)\,n\,v_{(i+1,j)} + (1-m)\,(1-n)\,v_{(i+1,j+1)} + m\,(1-n)\,v_{(i,j+1)}, \tag{3.30}$$

where m and n are the fractional positions within the current cell, as illustrated in Figure 3.5. The partial derivatives are bilinearly interpolated as follows,

$$\frac{\partial u}{\partial x} = (1-n)\left(\frac{mu_{(i,j+1)} - (1-m)u_{(i+1,j+1)}}{\Delta x}\right) + n\left(\frac{mu_{(i,j)} - (1-m)u_{(i+1,j)}}{\Delta x}\right), \quad (3.31)$$

$$\frac{\partial u}{\partial y} = m\left(\frac{nu_{(i,j)} - (1-n)u_{(i,j+1)}}{\Delta y}\right) + (1-m)\left(\frac{nu_{(i+1,j)} - (1-n)u_{(i+1,j+1)}}{\Delta y}\right). \quad (3.32)$$

The partial derivatives for v will be similar.

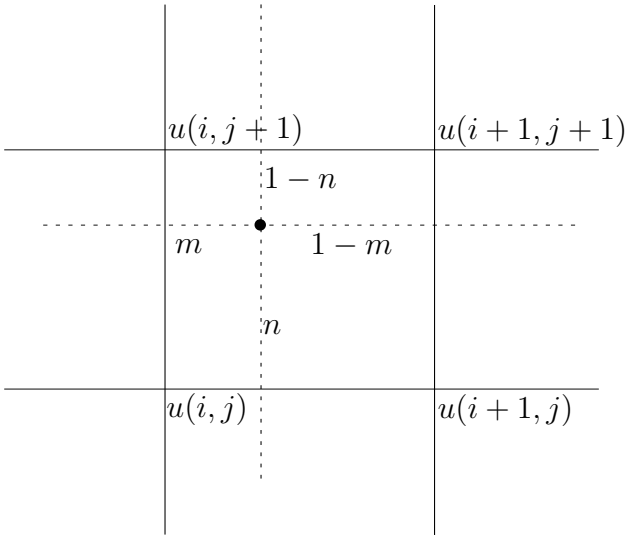


Figure 3.5: A visual representation of the fractional positions, m and n, within the grid cell, used to interpolate the velocity at any given point.

3.3.3 SINCEM algorithm

The SINCEM algorithm is an algorithm that was developed at the Laboratorio di Simulazione Numeriche del Clima e degli Ecosistemi Marini (SINCEM) by Fabbroni [2009]. Similar to TRACE, the SINCEM algorithm bilinearly interpolates the velocities in both the x and y directions, u, v such that,

$$u_p^k = m\,n\,u_{(i,j)} + (1-m)\,n\,u_{(i+1,j)} + (1-m)\,(1-n)\,u_{(i+1,j+1)} + m\,(1-n)\,u_{(i,j+1)}, \quad (3.33)$$

$$v_p^k = m\,n\,v_{(i,j)} + (1-m)\,n\,v_{(i+1,j)} + (1-m)\,(1-n)\,v_{(i+1,j+1)} + m\,(1-n)\,v_{(i,j+1)}, \quad (3.34)$$

where k indicates the current time step, and m and n are the fractional position of the particle within the cell, as in Figure 3.5. The SINCEM algorithm interpolated the velocity linearly in time.

The particles are labelled "lost" when they reach the boundaries of the numerical domain [Fabbroni, 2009]. For the purpose of this study, the algorithm only tracks particles in two dimensions.

3.3.4 SINCEM2 algorithm

SINCEM2 was also developed at the Laboratorio di Simulazione Numeriche del Clima e degli Ecosistemi Marini but is an improved method to determine the new particle location without having to determine a separate velocity at the particle's position by making use of the fourth-order Runge-Kutta integration method. The Runge-Kutta coefficients for the x direction are defined as,

$$q_1 = u(t^k, x_p^k)\,\Delta t, \tag{3.35}$$

$$q_2 = u(t^k + \frac{1}{2}\Delta t, x_p^k + \frac{1}{2}q_1)\,\Delta t, \tag{3.36}$$

$$q_3 = u(t^k + \frac{1}{2}\Delta t, x_p^k + \frac{1}{2}q_2)\,\Delta t, \tag{3.37}$$

$$q_4 = u(t^k + \Delta t, x_p^k + q_3)\,\Delta t, \tag{3.38}$$

where u is the zonal velocity components, Δt is the timestep and x_p^k is the particle location at time t^k. The final position of the particle is expressed as a combination of the previous position and a weighted average of the four coefficients,

$$x_p^{k+1} = x_p^k + \frac{q_1}{6} + \frac{q_2}{3} + \frac{q_3}{3} + \frac{q_4}{6}. \tag{3.39}$$

Equivalent relationships are obtained for the other directions as well.

A linear time interpolation is used to find the velocity field at the current timestep. The same boundary controls apply to SINCEM2 [Fabbroni, 2009].

3.4 Particle tracking results without diffusion

A trial run can be computed with the following set-up with an infected cage and an uninfected cage downstream.

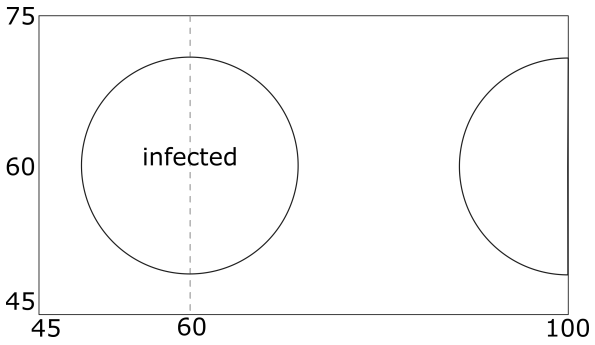


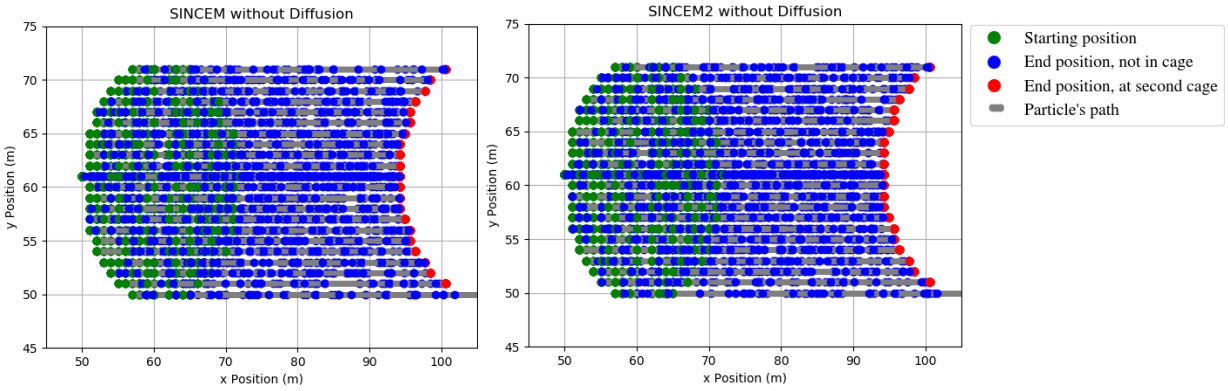
Figure 3.6: Set-up of a two cage system with a constant velocity in the x direction and a 0 velocity in the y direction.

The trial runs began with the particle field tests, where the particle clusters were generated within the infected cage as dictated by the population model. The parameters were set as listed in Table 3.2.

Table 3.2: Tabulated variables for a trial run of the particle tracking algorithm without diffusion in a constant velocity field.

Variable	Description	Unit
dt	1	s
T	1000	s
N_x	250	-
N_y	250	-
(x_0, x_{N_x})	x domain	m
(y_0, y_{N_y})	y domain	m
u_0	0.08	ms^{-1}
u_{N_x}	0.08	ms^{-1}
v_0	0.08	ms^{-1}
v_{N_y}	0.08	ms^{-1}
Cluster size	1×10^{10}	-

Figures 3.7a and 3.7b are the results for SINCEM and SINCEM2 respectively. The green dots represent the starting points, the blue dots represent the final positions of particles that do not reach the downstream cage, and the red dots represent the final positions of the particles that do reach the downstream cage. When a particle reaches the cage boundary, the particle position is no longer updated (this will not be the case in the final algorithm).



(a) SINCEM (b) SINCEM2

Figure 3.7: Test runs using the SINCEM and SINCEM2 algorithms on constant velocity fields in both the x and y directions, where the velocity in the y direction was $0\,\mathrm{ms}^{-1}$.

All of the cases had a total of 4385 clusters that were tracked. The simulation was run seven times for each case and the results are tabulated below. Table 3.3 shows that both algorithms give a relatively similar distribution of particle clusters (where the virion cluster is tracked as a single particle) that reach the second cage location, given that the initial locations of the particle clusters are chosen at random.

Table 3.3: Table illustrating the distribution of particle clusters that either reach the second, downstream cage or do not, within 1000 min. Two different algorithms were used and particle clusters had randomly generated locations within an initially infected cage. This illustrates the similarity in results with the two algorithms, SINCEM and SINCEM2.

SINCEM		**SINCEM2**	
Reached	Did not reach	Reached	Did not reach
845	3540	838	3547
817	3568	826	3559
862	3523	865	3520
794	3591	815	3570
830	3555	830	3555
793	3592	810	3575
820	3565	814	3571
18.77%		18.89%	

The results of this simulation without diffusion were as expected on a constant velocity field. The discrepancy in the results is likely due to the random assignment of starting positions within the initial cage.

3.5 Diffusion in particle tracking

A particle suspended within a fluid is likely to experience the effects of the diffusion of the fluid in which it is suspended. This additional diffusion term, and the derivation of the diffusion term, will be discussed in this section.

The velocity of the particle, where the particle is a cluster of virions,

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{v}(\mathbf{x}_p, t), \tag{3.40}$$

which was modelled in Section 3.3 and was solved through numerical integration. The velocity of the particle is comprised of a deterministic advection term and a diffusion term [Fabbroni, 2009]. This is described with a non-linear Langevin equation in each direction. The Langevin equations are stochastic differential equations where the stochastic dependence is linear [Gardiner, 1983; Toral, 2014]. The Langevin equation for each direction can be written as,

$$\frac{dx_i(t)}{dt} = a_i(x,t) + B_i(x,t)\zeta_i(t), \tag{3.41}$$

where $a_i(x,t)$ is the deterministic part of the motion, the $B_i(x,t)$ term signifies the random motion, and the $\zeta_i(t)$ is a random positive number between 1 and 0, and $i = x, y, z$, such that the component of each vector represents the three dimensions. Equation (3.41) may be rewritten as a system of linear equations as follows:

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{a}(\mathbf{x},t) + \mathbf{B}(\mathbf{x},t)\boldsymbol{\zeta}(t), \tag{3.42}$$

where $\mathbf{B}(\mathbf{x},t)$ is a matrix and $\boldsymbol{\zeta}(t)$ is a vector of independent random numbers.

Let,

$$dW_i = \zeta_i \, , dt \tag{3.43}$$

$$W_i(t) = \int_0^t \zeta_i(s)\, ds, \tag{3.44}$$

where W is the Wiener process in each direction, i.e. $i = x, y, z$. This process describes the random motion of a particle and is often referred to as a standard Brownian motion process. The Brownian motion process is achieved by accumulating the small random increments in dW for each direction. To achieve this, the small increment in equation (3.43) can then be substituted back into equation (3.42) to model the random diffusion of the particles

$$d\mathbf{x}(t) = \mathbf{a}(\mathbf{x},t)dt + \mathbf{B}(\mathbf{x},t)d\mathbf{W}(t). \tag{3.45}$$

The Wiener process has the following properties [Fabbroni, 2009; Toral, 2014]:

- The mean of the Wiener process is zero $\langle W(t) \rangle = 0$, and thus a normal distribution.

- The mean square value of the Wiener process is proportional to the timestep, $\langle dW dW \rangle = dt$ [Fabbroni, 2009; Toral, 2014].
- The Wiener process, $W(t)$ has a continuous path with a probability of 1 and thus happens almost surely.
- The Wiener process, $W(t)$, has independent increments.

Equation (3.45) may be discretised so that at each timestep, the diffusive term, can be added to the motion at each timestep,

$$\Delta \mathbf{x}(t) = \mathbf{a}(\mathbf{x}, t)\Delta t + \mathbf{B}(\mathbf{x}, t)\Delta \mathbf{W}(t). \tag{3.46}$$

The term $\Delta \mathbf{W}(t)$ is then replaced with a vector of independent random numbers $\boldsymbol{\mathcal{Z}}$ multiplied by the square root of the change in time $\sqrt{\Delta t}$. The vector of random numbers $\boldsymbol{\mathcal{Z}}$ has a standard normal distribution where the random numbers are between 1 and 0.

To solve for the two unknowns, $\mathbf{a}(\mathbf{x}, t)$, $\mathbf{B}(\mathbf{x}, t)$ the Fokker-Planck equation associated with equation (3.46) is solved. Where the Fokker-Planck equation governs the probability density of the Brownian motion,

$$\Delta \mathbf{x}(t) = \begin{bmatrix} u(x,t) \\ v(y,t) \\ w(z,t) \end{bmatrix} \Delta t + \begin{bmatrix} \sqrt{2\mathcal{D}_x} & 0 & 0 \\ 0 & \sqrt{2\mathcal{D}_y} & 0 \\ 0 & 0 & \sqrt{2\mathcal{D}_z} \end{bmatrix} \begin{bmatrix} \mathcal{Z}_1 \\ \mathcal{Z}_2 \\ \mathcal{Z}_3 \end{bmatrix} \sqrt{\Delta t}, \tag{3.47}$$

where the coefficients $\mathcal{D}_i$ $(i = \{x, y, z\})$ are the diffusion coefficients in the x, y, z directions, and $\mathcal{Z}_i$ $(i = \{1, 2, 3\})$ are independent random numbers, where $0 \leq \mathcal{Z}_i \leq 1$.

This method is then implemented by adding the following terms at each timestep

$$\mathcal{Z}_1 \sqrt{2\mathcal{D}_x \Delta t} \tag{3.48}$$

$$\mathcal{Z}_2 \sqrt{2\mathcal{D}_y \Delta t} \tag{3.49}$$

$$\mathcal{Z}_3 \sqrt{2\mathcal{D}_z \Delta t} \tag{3.50}$$

where,

$$\mathcal{Z}_1 \sqrt{2\mathcal{D}_x \Delta t} = [2\,\mathrm{randn}(0,1) - 1]\sqrt{2K_H \Delta t} \tag{3.51}$$

$$\mathcal{Z}_2 \sqrt{2\mathcal{D}_y \Delta t} = [2\,\mathrm{randn}(0,1) - 1]\sqrt{2K_H \Delta t} \tag{3.52}$$

$$\mathcal{Z}_3 \sqrt{2\mathcal{D}_z \Delta t} = [2\,\mathrm{randn}(0,1) - 1]\sqrt{2K_V \Delta t} \tag{3.53}$$

where $\mathrm{randn}(0, 1)$ is a normally distributed random number between 0 and 1, the function used in NumPy is `numpy.random.normal` which takes in a mean, standard deviation and size of the output vector. K_H and K_V are the horizontal and vertical diffusivity coefficients, i.e. $K_H = \mathcal{D}_x = \mathcal{D}_y$ and $K_V = \mathcal{D}_z$ [Fabbroni, 2009].

3.6 Particle tracking with diffusion preliminary results

Implementing the diffusion from Section 3.5 in the particle cluster's motion, the diffusion term from equations (3.51) and (3.52),

$$\mathcal{Z}_i\sqrt{\Delta t K_H}, \tag{3.54}$$

where $i = 1, 2$ are the directions in which diffusion is experienced. Equation (3.54) is then added to the change in the particle cluster's position at each timestep. The particle cluster is then referred to as a particle for simplicity. The final particle position at each timestep is then,

$$x_p^{k+1} = x_p^k + u_p^k\Delta t + \mathcal{Z}_1\sqrt{\Delta t K_H}, \tag{3.55}$$

in the x direction and the equivalent equation in the y direction is,

$$y_p^{k+1} = y_p^k + v_p^k\Delta t + \mathcal{Z}_2\sqrt{\Delta t K_H}. \tag{3.56}$$

Salama and Murray [2013] adds a tidal excursion to the particle tracking model, as this study will consider the influence of the fish cage, the tidal motion will also be affected by the presence of the cage. The tidal excursion is not considered in the particle tracking model.

The inputs were then chosen as follows [Salama and Murray, 2013],

$$K_H = 1\,\mathrm{m}^2/\mathrm{s}, \tag{3.57}$$
$$u = 0.08\,\mathrm{m/s}, \tag{3.58}$$
$$v = 0\,\mathrm{m/s}. \tag{3.59}$$

The SINCEM algorithm is considered, as the velocity field that will be considered in Chapter 6 is stationary. The random number generator used was the function `numpy.random.normal`, in Python, with a mean of 0 and a standard deviation of 0.3 to create a Gaussian distribution around 0 and between -1 and 1.

Fabbroni [2009] constructs a vector,

$$\boldsymbol{\mathcal{Z}} = \begin{bmatrix} \mathcal{Z}_1 \\ \mathcal{Z}_2 \\ \mathcal{Z}_3 \end{bmatrix}, \tag{3.60}$$

where the set of the random vectors is Gaussian distributed. This study is considering a two-dimensional system, thus a two-dimensional $\boldsymbol{\mathcal{Z}}$ vector,

$$\boldsymbol{\mathcal{Z}} = \begin{bmatrix} \mathcal{Z}_1 \\ \mathcal{Z}_2 \end{bmatrix}. \tag{3.61}$$

The random Gaussian distributed numbers are selected at each timestep ensuring that two particles released at the same position at the same time, do not follow the same path.

The algorithm was then run for 4 minutes of simulation time, the results are compared in Figures 3.8 and 3.9 to illustrate the random walk that the particle experiences.

The particle in Figure 3.8 travels approximately 2.5 m further than the particle in Figure 3.9 due to diffusion.

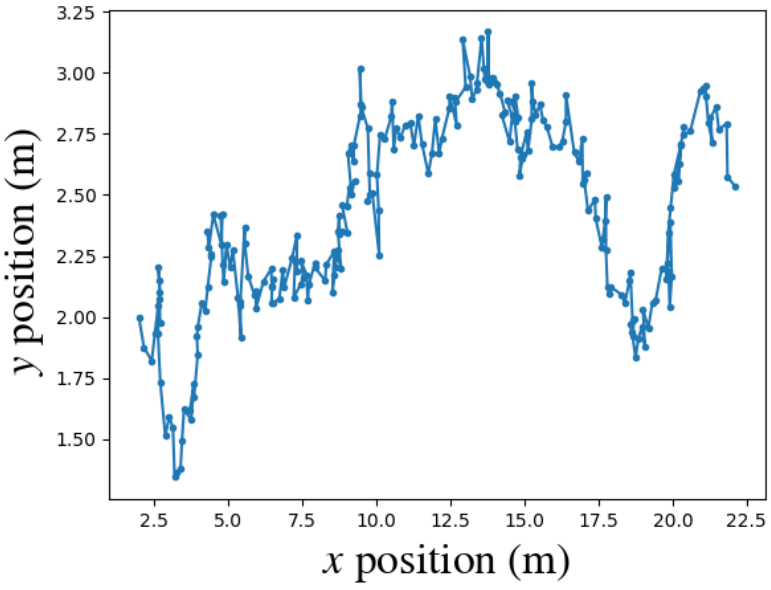


Figure 3.8: The first particle tracked with an initial position of $(2, 2)$. The particle is tracked with the SINCEM algorithm with diffusion, for a total time of 4 minutes and the variables in equations (3.57) to (3.59).

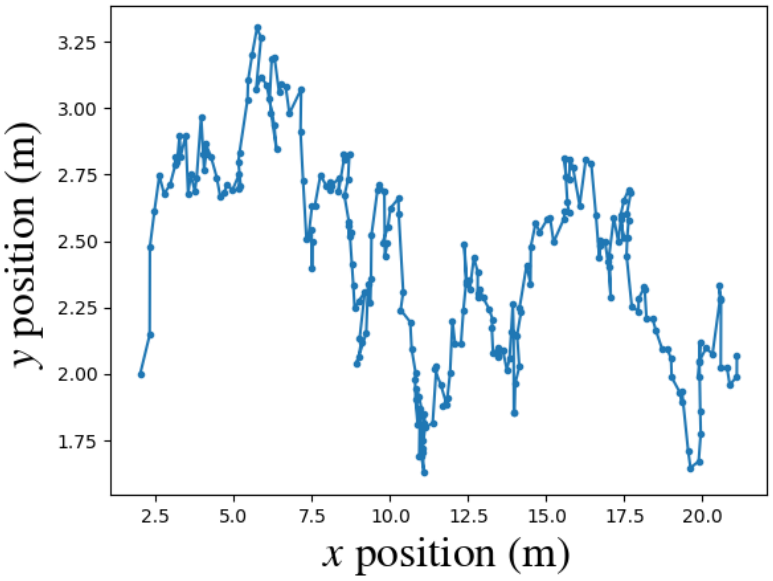


Figure 3.9: The first particle tracked with an initial position of $(x, y) = (2, 2)$. The particle is tracked with the SINCEM algorithm with the same parameters as Figure 3.8.

The particle in Figure 3.10 has a simulation time of 10 seconds, and illustrates the effect that diffusion has on the micro movements of the particle.

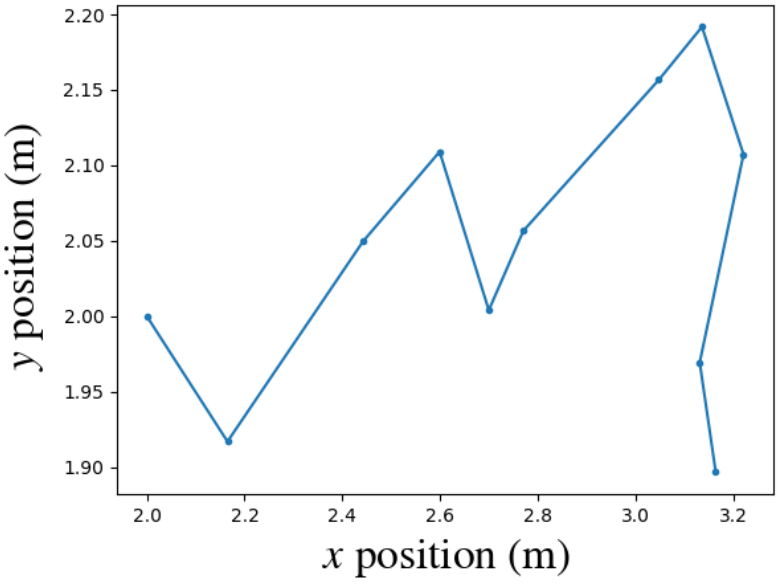


Figure 3.10: The first particle tracked with an initial position of $(x, y) = (2, 2)$. The total time for the simulation was 10 s.

In Figure 3.11, five particles were then tracked over a simulation period of one hour, all particles started at the same position and the final position of each particle was different.

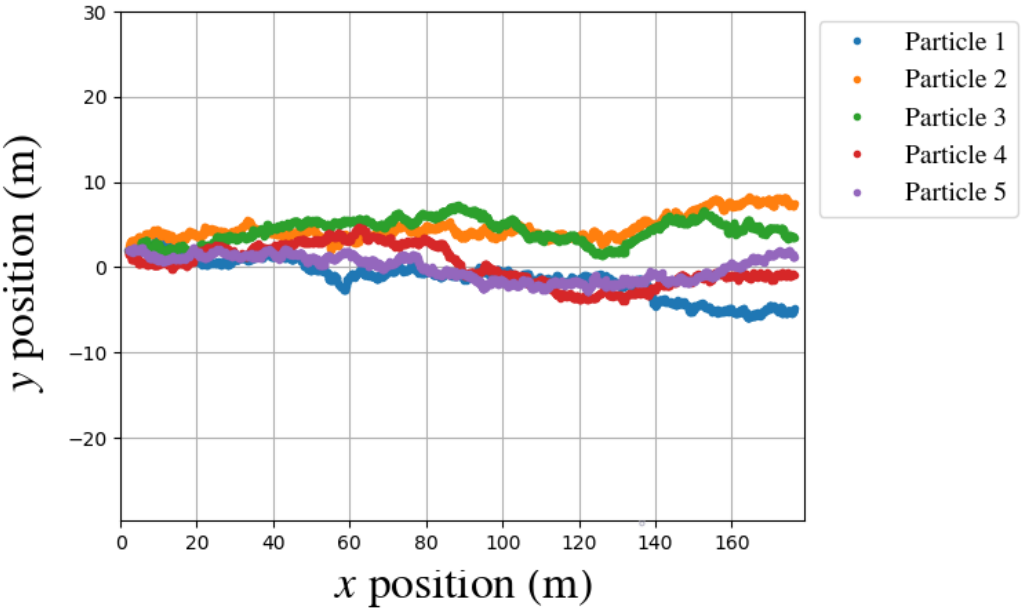


Figure 3.11: Five particles starting at $(2, 2)$ were tracked until they reached the boundary at $x = 177$, with $\Delta t = 1\,\text{s}$.

It should be noted that in all of these cases, the same stopping criteria were used, i.e. if either the particle reaches the x or y grid limits of 177 m and 122 m respectively, or if the predefined total time is reached, the particle will stop advancing. This is dependent on the geometry defined, it can easily be changed in the simulation script.

3.7 Reconsidering initially exposed E_0

The developed particle tracking model in Section 3.3 and the diffusion model in Section 3.5 are used to compare two extreme cases of the initially exposed population considered in Section 2.4. These are the cases where $E_0 = 0.8N$ and $E_0 = 0.1N$. The simulation time of the population model is of 140 days, as it takes approximately 100 to 140 days for the infection to clear. In Chapter 6 the particle tracking model will be run for a mere fraction of the approximate time it takes for the infection to clear. In Section 3.8 the decay of the virus is also considered.

The particle tracking was run for varying initial values of the initial exposed, E_0, with the parameters listed in Table 3.4.

Table 3.4: Tabulated variables for a trial run of the particle tracking algorithm with diffusion in a constant velocity field.

Variable	Description	Unit
dt	1	s
T	1000	s
Grid size, N_x	250	-
Grid size, N_y	250	-
(x_0, x_{N_x})	x domain	m
(y_0, y_{N_y})	y domain	m
u_0	0.08	ms^{-1}
u_{N_x}	0.08	ms^{-1}
v_0	0.08	ms^{-1}
v_{N_y}	0.08	ms^{-1}
Cluster size	1×10^{10}	-

In Figure 3.12, where the initial exposed population was 80% of the total population, there were a total of 4 385 particle clusters tracked. In the case of Figure 3.13, where the initial exposed population was 10% of the total population, only 612 particle clusters were tracked. In both Figures 3.12 and 3.13, the green dots represent the starting points, the blue dots represent the final positions of particles that do not reach the downstream cage, and the red dots represent the final positions of the particles that reach the downstream cage.

In Figure 3.12 it is clear that the second cage will become infected as 25% of the virions reach the second cage, whereas the case in Figure 3.13 only 0.2% of the particle clusters reach the second cage. The comparison between Figures 3.12 and 3.13 illustrates the importance of choosing the percentage of fish initially exposed to the virus. It is vital to ensure that the number of fish that are initially exposed truly represents the dynamics of the system, as it can lead to an under or over prediction of the probability of the spread of disease downstream.

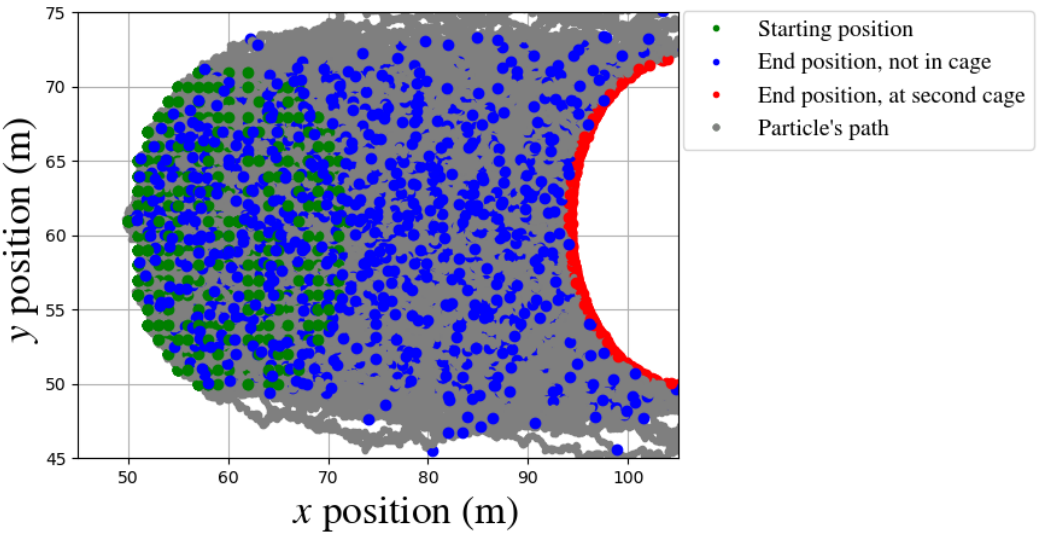


Figure 3.12: Virions with randomly generated initial positions where the population model had an initially exposed population of $E_0 = 0.8N$. A constant velocity field was used with a diffusion term, with parameters that describe the ISAV-type, 1103 particle clusters reached the second cage and 3282 particle clusters did not.

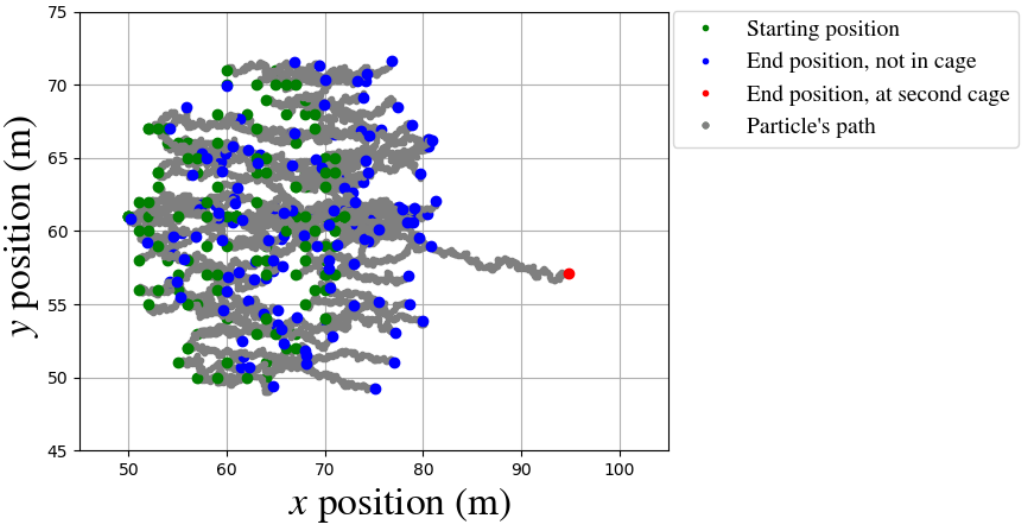


Figure 3.13: Virions with randomly generated initial positions where the population model now had a lower initially exposed population of $E_0 = 0.1N$. A constant velocity field was used with a diffusion term, with parameters that describe the ISAV-type, only 1 cohort reached the second cage and 611 particle clusters did not.

For the various cases of the initially exposed population as a percentage of the total population, the number of particles shed over 140 days can be calculated

and are tabulated in Table 3.5.

Along with Figures 3.12 and 3.13, Table 3.5 illustrates the significance of appropriately choosing the percentage of initially exposed population as the number of particles shed is approximately one order of magnitude larger for the $E_0 = 99\%N$ exposed population than the $E_0 = 10\%N$ exposed population, which is expected.

Table 3.5: Tabulated results of the total virions shed over a period of 150 days, with parameters consistent with that of the ISAV-type.

Size of E_0 as a percentage of the total number of fish	Number of virions
10%	2.14×10^{19}
50%	8.63×10^{19}
80%	1.23×10^{20}
99%	1.43×10^{20}

3.8 Shedding and decay of virions

Biological particles will not always function at full efficiency over time and often replicate. Thus the process of shedding and decay of virions in this system must be considered. For this study only virions will be considered, and not bacteria, as is stated in Section 3.2, which allows for a simplistic shedding and decay model.

The virions are replicated through a process of shedding from an infected fish, i.e. a virion that is outside of the fish cannot replicate. The number of virions shed during the infection is proportional to the number of infected fish at each timestep, as each fish will shed at a constant rate depending on the biological properties of the virus in question. Each virus will shed at a different rate Γ. The number of virions released at each timestep can be described by the linear equation,

$$\mathcal{W}_{k+1} = \Gamma I_k, \tag{3.62}$$

where $\mathcal{W}_{k+1}$ is the waterphase, which is the number of new virions that will be present in the simulation at the next timestep. The term Γ is the shedding rate and I_k is the number of infected fish at the previous timestep. The new virions are clustered according to the cluster size and then each cluster is considered as a single particle. Equation (3.62) makes use of the number of infected fish at the previous timestep, as the new particles at each timestep will be assigned a starting position and must be tracked along with the remainder of the particles in each timestep. Virions are not alive [Microbiology Society, 2016], and thus rely on the cells of the host animal to replicate, so the production of particles is limited to the infected cage boundaries. The particles also cannot die as they

are not alive, the particles can, however lose effectiveness at infecting a new host. This process will be referred to as particle decay. The effectiveness of all of the particles is dependent on the time that has passed since each particle was created. The function of decay is exponential and can be described by the exponential decay equation,

$$\mathcal{W}_{k+1} = \Gamma I_k e^{-\lambda t}, \tag{3.63}$$

where λ is the decay coefficient.

In practice the number of new particles will first be calculated, the new and old particles are given a new position and then the decay is considered, breaking equation (3.63) into two separate equations. Once decay is considered, any particle that has lost effectiveness is discarded entirely from the model. This ensures that ineffective particles do not take up memory within the simulation.

The virus considered is infectious salmon anaemia virus (ISAV). ISAV sheds at a rate of $7.2 \times 10^{-1}\,\mathrm{ml^{-1}h^{-1}kg^{-1}}$ and decays at a rate of $0.12\,\mathrm{h^{-1}}$ [Salama and Murray, 2013][Salama and Murray, 2011].

For a cage consisting of 9×10^4 mature Atlantic salmon, with each salmon weighing 4.5 kg [Britannica, 2019], the total weight of the biomass is 40.5 t. The cage size considered has a 30 m diameter and a depth of 13 m [Winthereig-Rasmussen et al., 2016], the volume of the cage is then $1225\,\mathrm{m^3}$. The shedding rate then used for the current simulation was 200 virions per second per kilogram per metre cubed, as calculated below.

$$\begin{aligned}\text{shedding rate}(\mathrm{s^{-1}kg^{-1}m^{-3}}) &= \text{shedding rate}(h^{-1}kg^{-1}ml^{-1})\frac{1}{(3600)(1\times 10^6)}\\ &= 7.2\times 10^{-1}\frac{1}{(3600)(1\times 10^6)}\end{aligned}$$

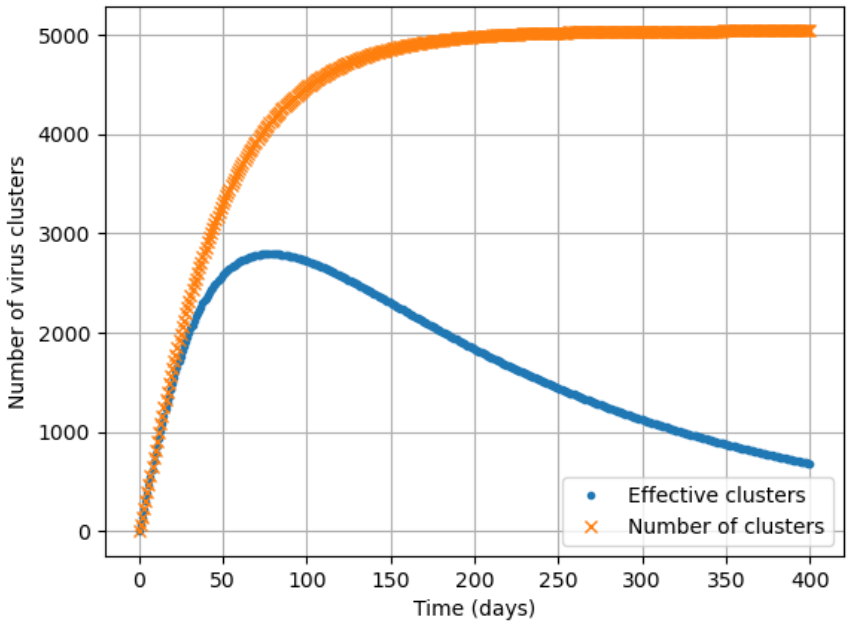


Figure 3.14: ISAV-type particles after 400 days of the infection, where both the total particles and the particles remaining after decay are plotted.

In Figure 3.14 the total particles shed for the 40.5 t cage were plotted as well as the decay to illustrate the time it takes for the ISAV particles to lose efficiency, and thus how many particles are capable of infecting an exposed fish. The cluster size was kept at 5×10^{12} and the timestep was set to 1 day.

3.9 Minimum infection threshold

The minimum infection threshold can be defined as the minimum number of virions needed per cubed metre to infect a fish that passes through the area. The value for minimum infection threshold, φ, given in the literature is $10\,\mathrm{ml}^{-1}\,\mathrm{kg}^{-1}$[Gregory et al., 2009] which was calculated by dividing by the mass of fish and volume in the experimental tanks. This value is therefore dependent on the density of fish in the vicinity and will be different within and outside of the boundaries of the cage.

The TCID_{50} refers to the 50% tissue culture infectious dosage, which signifies the concentration at which 50% of the cells are infected. In virology a plaque-forming units, PFU is the number of virions necessary to rupture the membrane of a cell. As this study tracks a cluster of virions over a domain, the PFU is a more useful metric.

The conversion between TCID_{50} and PFU is approximately [PubMed, 2019],

$$\begin{aligned}\text{PFU} &= 0.69\,\text{TCID}_{50}\,\text{ml}^{-1}\,\text{kg}^{-1},\\ &= 0.69(1\times 10^{1})\,\text{ml}^{-1}\,\text{kg}^{-1},\\ &= 6.9\times 10^{6}\,\text{m}^{-3}\,\text{kg}^{-1}.\end{aligned}$$

The weight per cubed metre within the cage is,

$$\begin{aligned}\text{density} &= \frac{\text{weight of the captive fish}}{\text{cage volume}},\\ &= \frac{405000}{2\pi(15)^{2}(13)},\\ &= 22\,\text{kg}\,\text{m}^{-3}.\end{aligned}$$

If, however, only one fish swims through a square metre,

$$\begin{aligned}\text{density} &= \frac{\text{weight fish}}{\text{volume}},\\ &= \frac{4.5}{1},\\ &= 4.5\,\text{kg}\,\text{m}^{-3}.\end{aligned}$$

Given the density of fish per square metre in the respective areas, within the cage the infection threshold is $1.518\times 10^{8}\,\text{m}^{-3}$ PFU and outside the cage is $3.105\times 10^{7}\,\text{m}^{-3}$ PFU. A conservative approach is taken, and the smallest minimum threshold is used as the threshold both within and outside the cage boundaries.

Chapter 4

Influence of the fish cage

If a large object is placed within a velocity field, the object may distort the velocity field significantly. The distortion of the velocity field is dependent on the size, roughness, shape and porosity of the object as well as the properties of the velocity field. As a fish cage is essentially a large obstacle within a velocity field, it is realistic to assume that the cage will distort the velocity field in which it is placed.

Salmon are farmed in large open water cages, approximately 30 m in diameter which span up to 13 m deep. The cages are generally placed in two rows, on a grid. The cage basically consists of a floating ring, a sinker ring, a cylindrical net and connective twine that holds it all together, as is illustrated in Figure 1.1. Only the net will be simulated as it is assumed that the other components have a negligible effect on the velocity field [Bi and Xu, 2018; Winthereig-Rasmussen et al., 2016]. The cage will be considered to be a porous medium when evaluating the cage's effect on the velocity field [Winthereig-Rasmussen et al., 2016].

Winthereig-Rasmussen et al. [2016] stated that the flow was generally well predicted within the cage but the flow reduction in the wake of the cage was over-predicted. The study also found that the fluid velocity was over-predicted by up to 50% which is then attributed by Winthereig-Rasmussen et al. [2016] to the cage deformation and the fish behaviour within the cage. Due to the complexity of modelling the deformation and bathymetry of the problem, these are not considered, as it was found that a 30% difference in solidity caused by deformation only lead to 10% difference in the drag coefficient of the net. It is then an appropriate assumption that the deformation does not have a significant influence on the cages modelled in this manner. Winthereig-Rasmussen et al. [2016] noted a better correlation between experimental and simulation results were observed at higher velocities, however, salmon cages are generally placed within relatively slow velocity fields in sheltered waters to minimise the risk of fish escaping into the wild surrounding areas and to potentially minimise damage to the cage caused by rough seas.

A cage that is placed within a natural environment is subject to the growth of algae, plants, microorganisms and molluscs on the net of the cage. This reduces the porosity of the net and is referred to as biofouling. Bi and Xu [2018] focused on the effect of biofouling on the porosity of the net in their study. Although, even with no biofouling, the velocity within the cage was found to reduce the incoming velocity significantly. The velocity reduction observed in the study by Bi and Xu [2018] did not align with experimental results obtained by Turner et al. [2015]. Turner et al. [2015] found the reduction in the velocity of the centreline of the cage, directly after the cage, was reduced so much that it was under the Swoffer meter (a Swoffer meter is a rod with a propeller equipped with a fibre-optic sensor that uses electric pulses to determine the velocity of a flow field) threshold and subsequently unmeasurable by the equipment, thus the flow was at most $0.2U_0$, where U_0 is the average velocity on the inlet boundary.

It is evident from the results of the various positions of the cages in Bi and Xu [2018], that the case considered in this study (two cages, one directly downstream from the other) is the worst-case scenario in terms of velocity reduction.

4.1 OpenFOAM simulations

As mentioned in Section 1.1, OpenFOAM is an opensource C++ toolbox used to simulate fluid dynamics systems.

An OpenFOAM simulation was run to determine the effects of the cage on the velocity field. The simulation was run with a Euler backwards time scheme, various turbulence models and a Newtonian transport model. Two separate turbulence models were used, a $k-\epsilon$ turbulence model and a realizable $k-\epsilon$ turbulence model respectively. The simulations were also run without a turbulence model. The resistance coefficients for these models were based on a study by Patursson [2008].

Similar to the study by Winthereig-Rasmussen et al. [2016], the model does not include bathymetry data and possible effects of a relatively flat bottom. The shoreline was also not taken into consideration. The problem is therefore reduced to essentially two dimensional simulations. The focus of this study is to simulate the effect of, amongst others, the flow through and around cages and their effects on the spread of the virus between cages. Therefore only a constant inlet flow boundary was considered and the extension to semidiurnal tidal currents are left for further research.

The six cases considered in this study were as follows and will henceforth be referenced by their case number:

- Case 1: One row, two cages per row, simulated with a $k-\epsilon$ turbulence model
- Case 2: One row, two cages per row, simulated with no turbulence model
- Case 3: Two rows, two cages per row, simulated with a $k-\epsilon$ turbulence model
- Case 4: Two rows, two cages per row, simulated with no turbulence model
- Case 5: One row, two cages per row, simulated with a realizable $k-\epsilon$ turbulence model
- Case 6: Two rows, two cages per row, simulated with a realizable $k-\epsilon$ turbulence model

Figure 4.1 shows the boundaries and the layout of the simulation to scale. The two cages, each with a radius of 15 m, have their centres at $(-50, 0)$ and $(50, 0)$ respectively. The left boundary is the inlet with an inlet velocity of $U_0 = 0.5\,\mathrm{ms}^{-1}$. This set-up is used for Cases 1, 2 and 5, respectively.

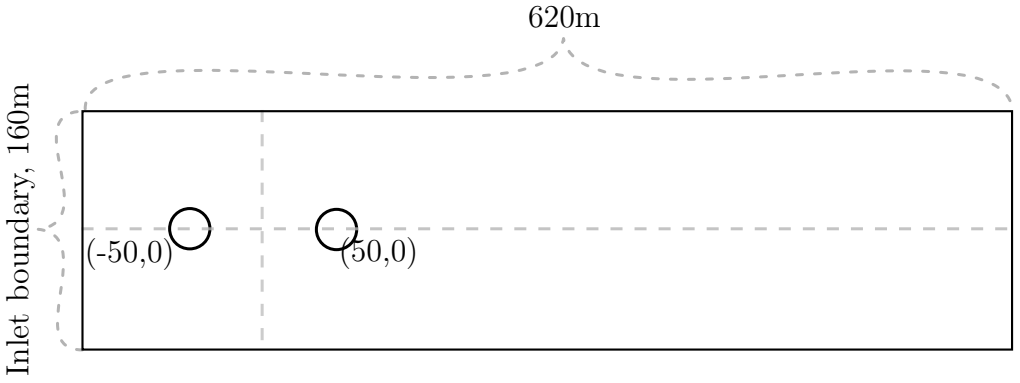


Figure 4.1: Cases 1, 2 and 5: The layout of one row of two simulated cages with an inlet boundary on the left, and two 15 m radius circular cages with centres at $(-50, 0)$ and $(50, 0)$ respectively. This figure is to scale.

Similar to the layout with one row of two cages, Figure 4.2 shows the boundaries and the layout of the simulation to scale. The four cages, each with a radius 15 m, have their centres at $(-50, 0)$ and $(50, 0)$, $(-50, 35)$ and $(50, 35)$ respectively. The left boundary is the inlet with an inlet velocity of $U_0 = 0.5\,\mathrm{ms}^{-1}$. This set-up is used for Cases 3, 4 and 6.

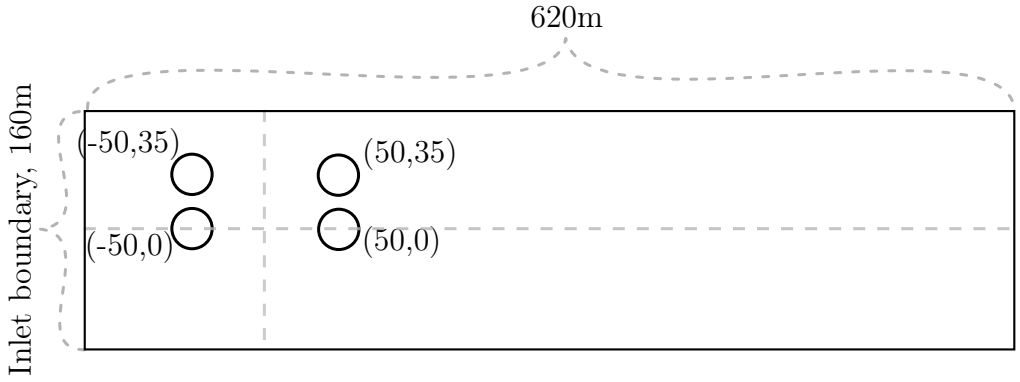


Figure 4.2: Cases 3, 4 and 6: The layout of two rows of two simulated cages with an inlet boundary on the left, and two 15m radius circular cages with centres at $(-50, 0)$, $(50, 0)$, $(-50, 35)$ and $(50, 35)$ respectively. This figure is to scale.

The porous media model was run with the cage modelled as a cylindrical tube, with a porosity of $\phi = 0.80$ [Winthereig-Rasmussen et al., 2016] where the corresponding resistance coefficients were found in Patursson [2008]. The resistance coefficient matrices used were,

$$D = \begin{bmatrix} D_n & 0 & 0 \\ 0 & D_t & 0 \\ 0 & 0 & D_t \end{bmatrix} = \begin{bmatrix} 51730 & 0 & 0 \\ 0 & 26379 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{4.1}$$

$$C = \begin{bmatrix} C_n & 0 & 0 \\ 0 & C_t & 0 \\ 0 & 0 & C_t \end{bmatrix} = \begin{bmatrix} 5.0980 & 0 & 0 \\ 0 & 1.6984 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{4.2}$$

where D and C are matrices of resistance coefficients. The coefficients D_n and C_n are the resistance coefficients normal to the porous media, and the coefficients D_t and C_t are resistance coefficients tangential to the porous media. These coefficients are defined within the OpenFOAM simulation and governing equations. The geometry was set up such that the wall of the cylindrical tube is a porous medium with the same porosity as the net of the cage.

The grid used in the set-up of this simulation is a non-uniform rectangular grid. The grid independence was checked by initially running the simulation with a coarse grid of $N_x \times N_y = 1240 \times 320$ and then with a fine grid of $N_x \times N_y = 2480 \times 640$. The two simulations both refined the mesh on the porous media. The mesh refinement made use of hex topology to refine the mesh on the porous media. The residuals were calculated by first spatially interpolating the velocity fields over the two grids at each timestep. Interpolation allows the two velocity fields to be compared at the same position in each direction.

By finding the difference between the velocity fields of the coarse and fine grid in each direction, two velocity fields, ΔU and ΔV, were created in the

x and y directions respectively. The relative magnitude was then calculated by calculating the sum of the squares of the two directions, and diving by the velocity magnitude of the coarse grid at the same position. The residuals, as percentages, are plotted in Figure 4.3. The maximum average relative difference is 4.55%, and the average relative difference is 2.12%. Figure 4.3 indicates that CFD simulation is independent of the grid used.

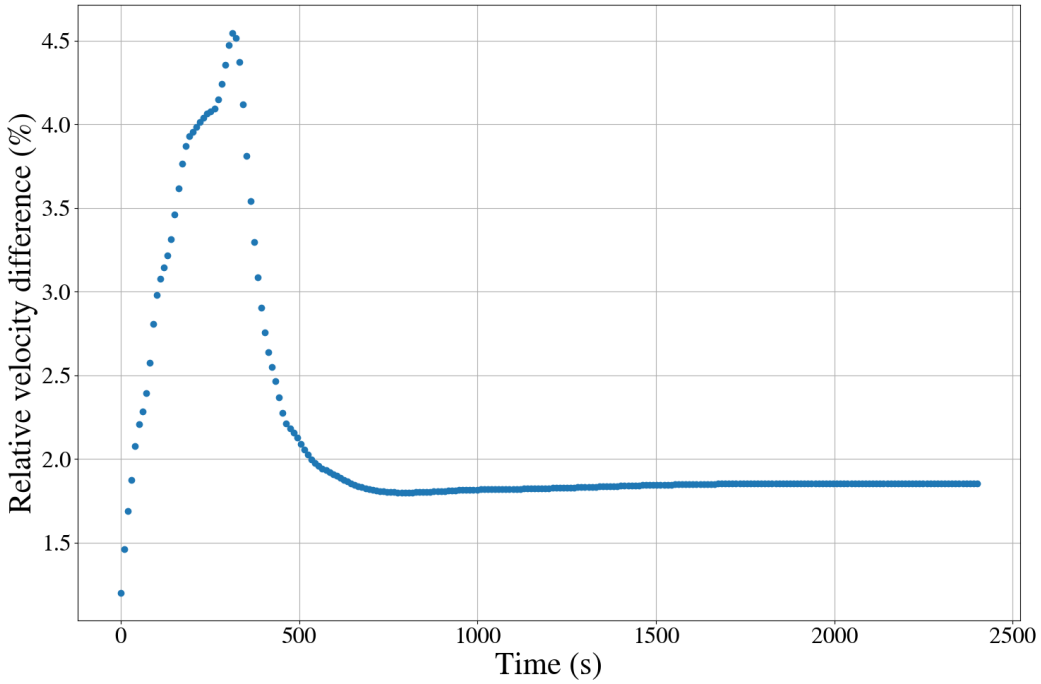


Figure 4.3: The residuals plotted between the coarse and fine grids to indicate grid independence.

One of the necessary conditions for convergence is the Courant number. The Courant number $\mathcal{C}$ is typically calculated for each cell and is defined as the absolute velocity in the cell, multiplied by $\frac{\Delta t}{\Delta x}$ where Δt is the timestep and Δx is the length of the cell. In this OpenFOAM simulation, the maximum Courant number was set to $\mathcal{C}_{\max} = 0.7$.

In practice, the porous media model should include the presence of the fish, with a second porous medium within the cylinder, with a porosity corresponding to that of a given volume of fish within a given cage volume. As the fish are in constant motion, a model of the disturbance caused by the fish should also be considered within the model. This is however, beyond the scope of this study and will not be considered.

The inlet boundary velocity of this simulation is $0.5\,\mathrm{ms}^{-1}$ and the diameter of the cage is 30 m. This leads to a large Reynolds number of 1.74×10^6, however,

the velocity field computed is not turbulent. An alternative Reynolds number that can be considered is that of the net, which was calculated to be 265. These Reynolds numbers are within the same range as those found by Turner et al. [2015].

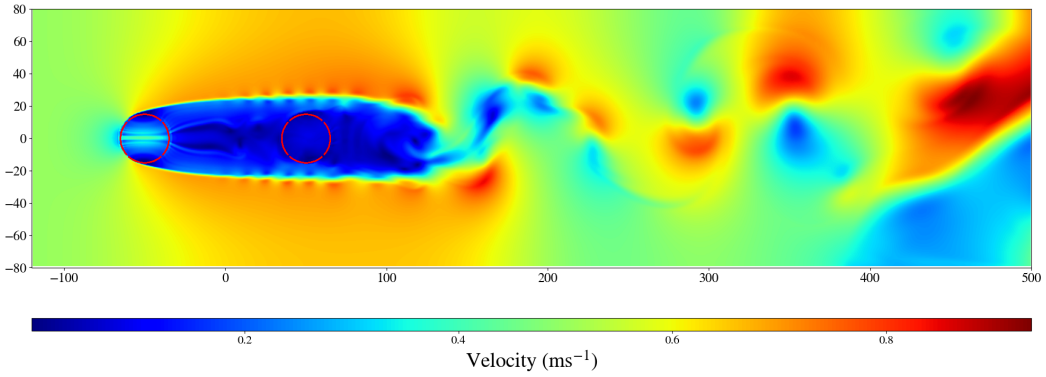


Figure 4.4: Case 2: Velocity magnitude around the two cages in succession after $t = 40$ min simulation time in OpenFOAM.

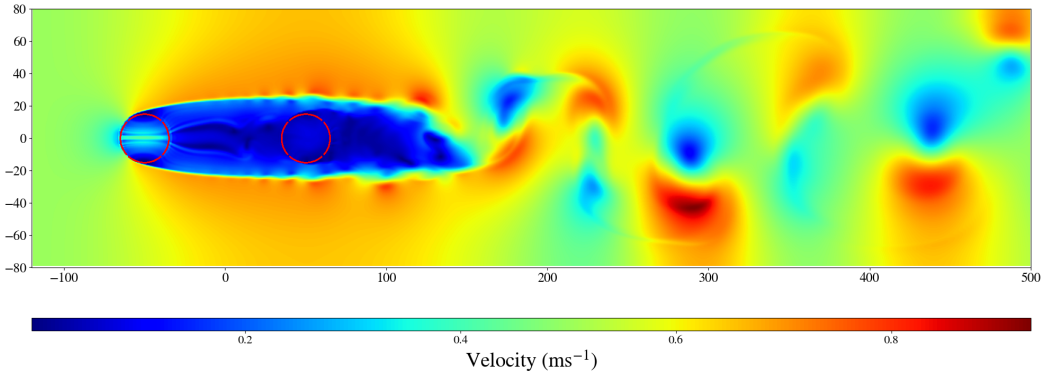


Figure 4.5: Case 2: Velocity magnitude around the two cages in succession after $t = 50$ min simulation time in OpenFOAM.

In both Cases 2 and 4, where no turbulence model was considered, vortex shedding is experienced. The velocity fields at various timesteps are plotted in Figures 4.4 and 4.5.

The vortex shedding observed at each of the timesteps in Cases 2 and 4 is expected as experimental results from Turner et al. [2015] displayed similar vortices behind the cages. It should be noted that in the experiment conducted by Turner et al. [2015], the cage comprised of two nets, an inner main net and an outer predator net, which may, to some extent, be accounted for by the thickness of the porous cylinder in this study.

Vortex shedding is a laminar phenomenon that occurs at Reynolds numbers between 40 and 2×10^5, where vortices are created periodically and predictably behind a body [Aerospace Engineering, 2016]. The area behind the cylinder that experiences these periodic vortices is known as the von Kármán vortex street. This phenomenon is discussed further in Appendix D.

The simulation was also run with a $k - \epsilon$ turbulence model. This removed the vortex shedding entirely. Similar results were observed when a realizable $k - \epsilon$ model was used in both Bi and Xu [2018] and Winthereig-Rasmussen et al. [2016]. Therefore the decision was made to include the realizable $k - \epsilon$ turbulence model as an additional case, as can be seen in the list earlier in this chapter.

As is illustrated by Figure 4.6 and Figure 4.7, fairly uniform flow patterns behind the cages are observed.

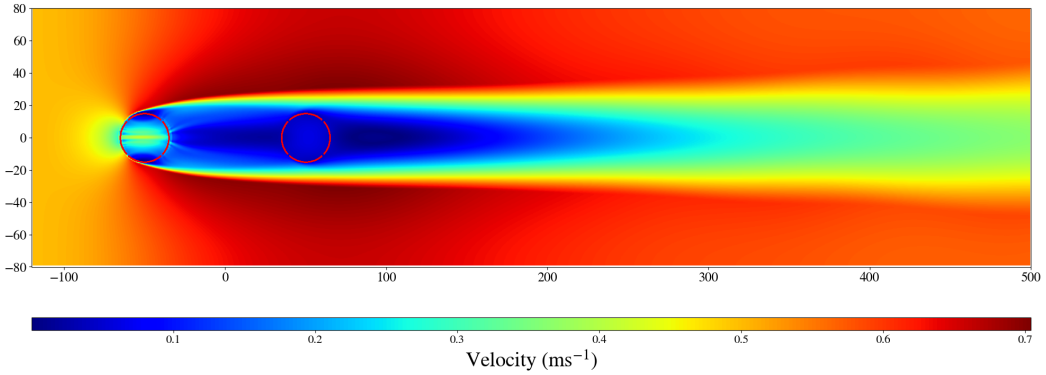


Figure 4.6: Case 1: Velocity magnitude around the two cages in succession with a $k - \epsilon$ model, simulated in OpenFOAM at $t = 40$ min.

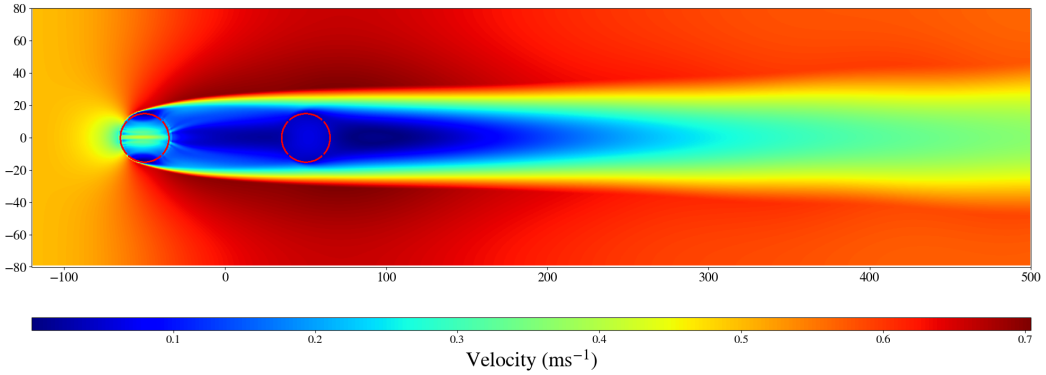


Figure 4.7: Case 1: Velocity magnitude around the two cages in succession with a $k - \epsilon$ model, simulated in OpenFOAM at $t = 50$ min.

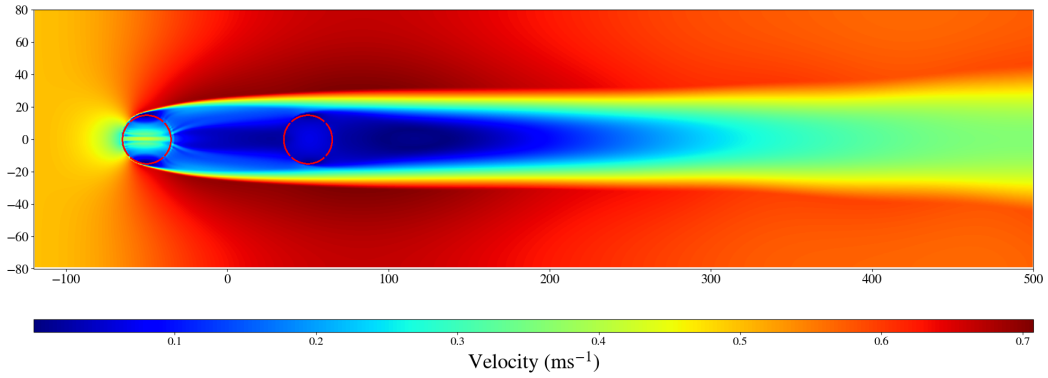


Figure 4.8: Case 5: Velocity magnitude around the two cages in succession with a realizable $k-\epsilon$ model, simulated in OpenFOAM at $t = 40\,\text{min}$.

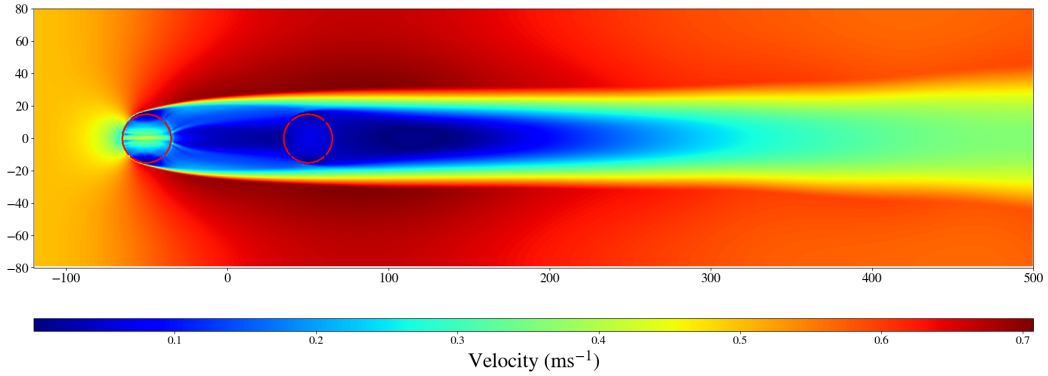


Figure 4.9: Case 5: Velocity magnitude around the two cages in succession with a $k-\epsilon$ model, simulated in OpenFOAM at $t = 50\,\text{min}$.

The simulated velocity fields at $t = 40\,\text{min}$ and $t = 50\,\text{min}$ are shown in Figure 4.8 and Figure 4.9 respectively. It is observed that, as is the case with the $k-\epsilon$ model, a similar uniform flow pattern is observed.

When simulated without a turbulence model, it is evident that there is vortex shedding, however, when a $k-\epsilon$ turbulence model or realizable $k-\epsilon$ turbulence model is used to simulate the situation, the flow pattern is fairly uniform. Both $k-\epsilon$ and realizable $k-\epsilon$ are Reynolds-averaged Navier-Stokes (RANS) two equation models [Wasserman, 2016]. The RANS models make use of Reynolds decomposition to solve the Navier-Stokes equations. Reynolds decomposition is a method of averaging with respect to time. This process of averaging the Navier-Stokes equations results in a smoother velocity field and the reduction of fluctuating quantities, and thus a lower resolution than more computationally expensive methods. Computationally expensive methods include large

eddy simulation (LES) which makes use of signal filtering techniques, and direct numerical simulation (DNS) which solves the Navier-Stokes equations at high resolution.

Although vortex shedding is evident in the experimental study, Turner et al. [2015], the present study investigates the significance of the vortex shedding and whether the averaged solution is a good approximation of the physical system.

4.2 Averaging the velocity fields over time

These simulations were conducted on a personal computer and in order to allow for repeatability of the simulations, the simulations were kept as computationally inexpensive as possible. Updating the velocity at each timestep in the particle tracking algorithm can be computationally expensive and therefore, a time averaged velocity field was used with the particle tracking model, instead of a time series. The time averaged velocity field, without a turbulence model, is similar to the velocity profile calculated with a $k-\epsilon$ turbulence model or a realizable $k-\epsilon$ turbulence model as well as those computed in Winthereig-Rasmussen et al. [2016] and Bi and Xu [2018]. The time averaged velocity field over 1 800 s (approximately 30 min) was computed for Case 2 (no turbulence) and is shown in Figure 4.10, with the velocity field streamlines.

The extent to which these average velocity fields are similar is investigated further in the following chapters.

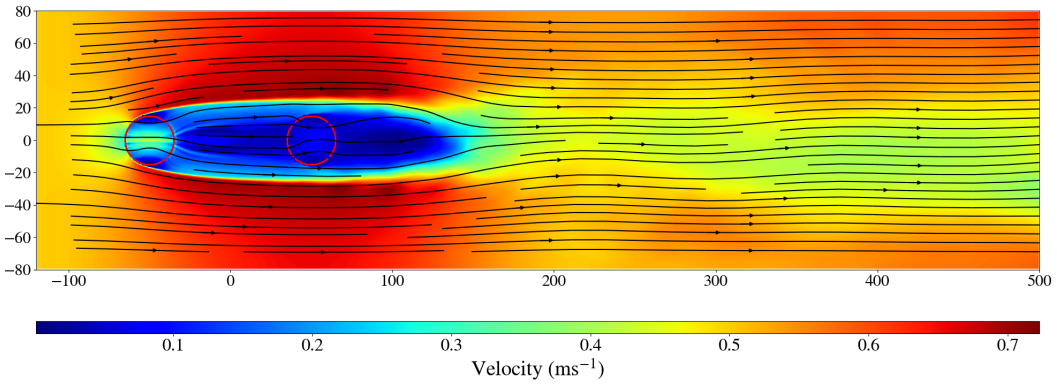


Figure 4.10: Case 2: Velocity magnitude of the time averaged velocity field simulated in OpenFOAM.

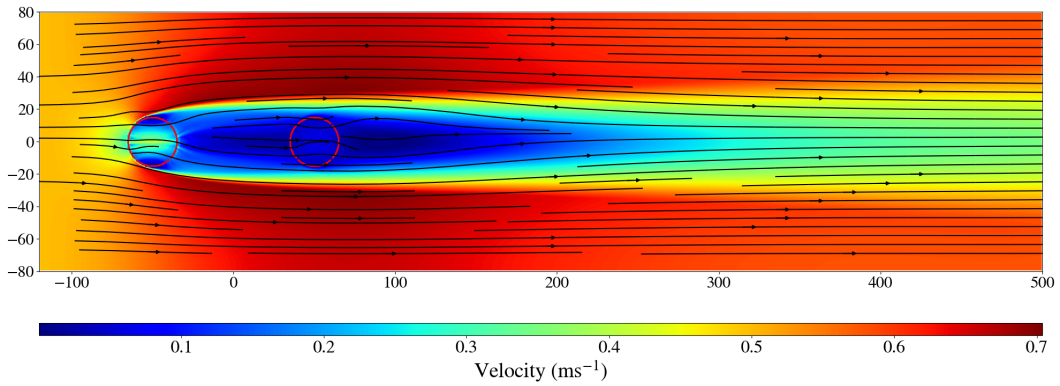


Figure 4.11: Case 1: Velocity magnitude of the time averaged velocity field simulated with a $k - \epsilon$ turbulence model in OpenFOAM.

In Figure 4.11 the averaged velocity field, with streamlines, for Case 1 ($k - \epsilon$ turbulence model) is shown. By comparing Figure 4.11 to Figures 4.6 and 4.7, it is apparent that the velocity fields exhibit similar patterns. The residuals for this case will be discussed further at the end of this section. The average velocity field, with streamlines, of Case 5 (realizable $k - \epsilon$ turbulence model) is shown in Figure 4.12, which resembles the velocity field of Case 1.

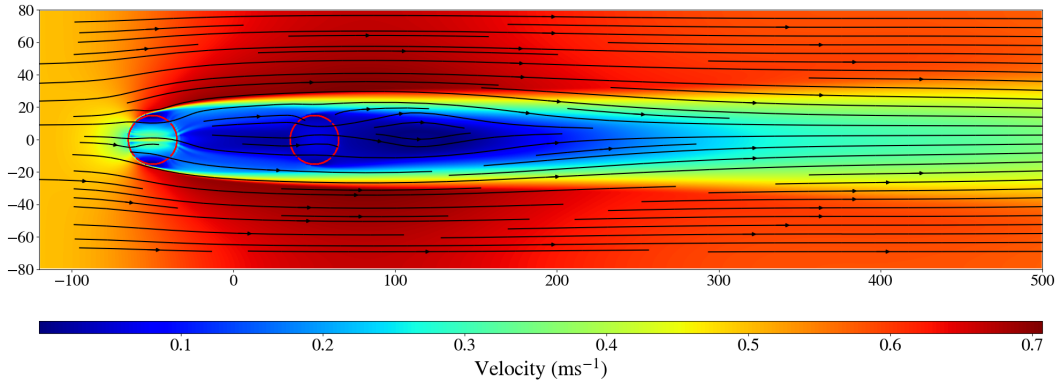


Figure 4.12: Case 5: Velocity magnitude of the time averaged velocity field simulated with a realized $k - \epsilon$ turbulence model in OpenFOAM.

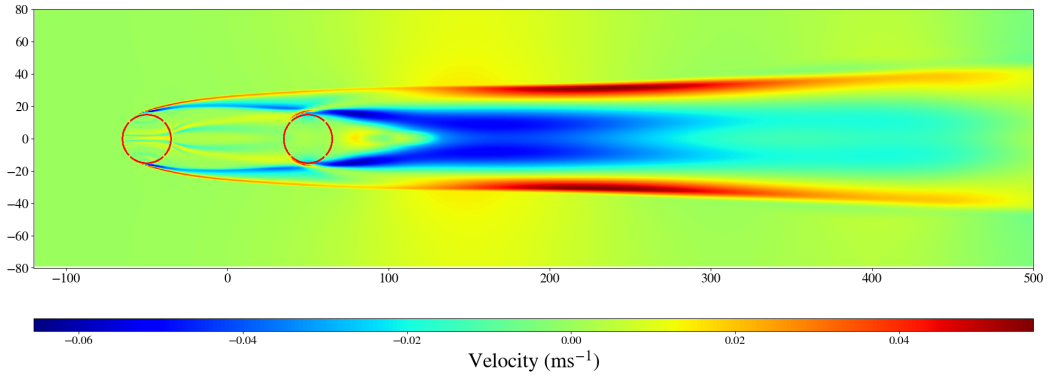


Figure 4.13: Case 1 subtracted from Case 5: Velocity magnitude of the resultant time averaged velocity field between the velocity fields simulated with a $k-\epsilon$ turbulence model and a realizable $k-\epsilon$ turbulence model in OpenFOAM.

The average velocity fields can also be used to compare the simulations done with a $k-\epsilon$ turbulence model and a realizable $k-\epsilon$ turbulence model, respectively. In Figure 4.13 the average velocity field of Case 5 is subtracted from the average velocity of Case 1. This is done by subtracting component-wise and finding the magnitude of the resultant. As expected, the resultant velocity field indicates that the two calculations are identical for all practical purposes, with a maximum deviation of approximately 10% of the incoming velocity.

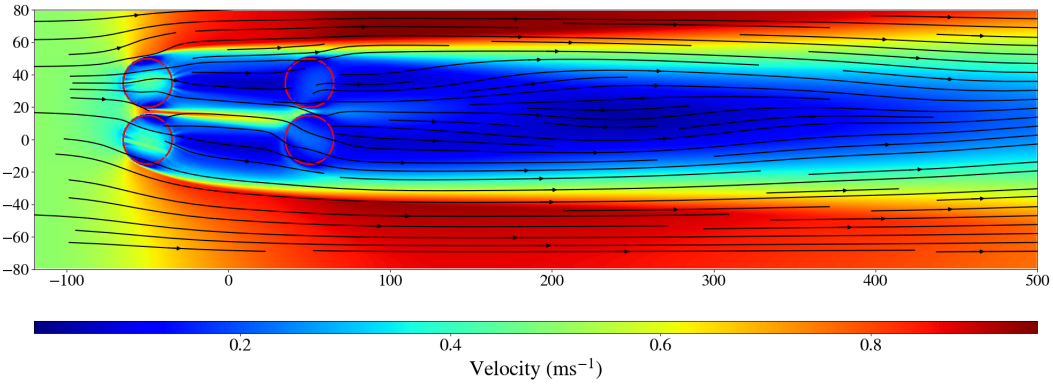


Figure 4.14: Case 3: Velocity magnitude of the time averaged velocity field simulated with a $k-\epsilon$ turbulence model in OpenFOAM.

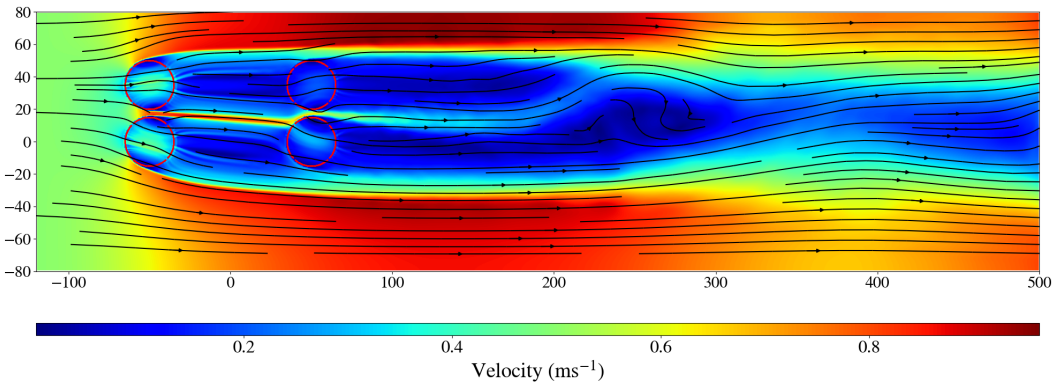


Figure 4.15: Case 4: Velocity magnitude of the time averaged velocity field simulated with a $k-\epsilon$ turbulence model in OpenFOAM.

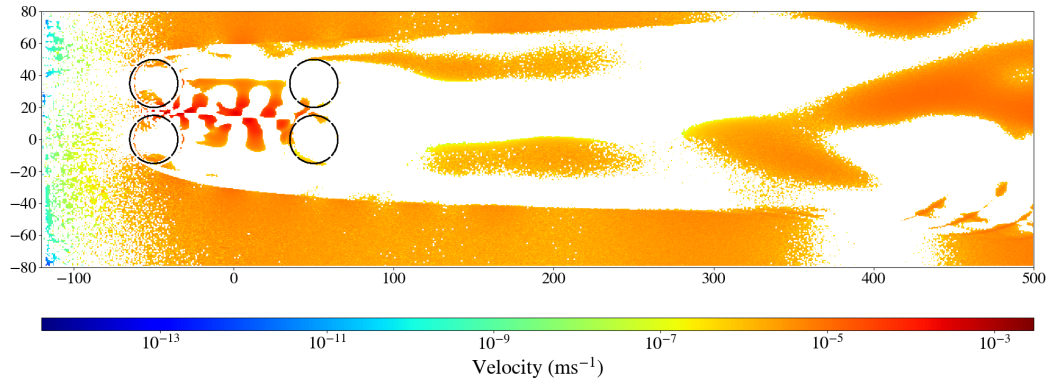


Figure 4.16: Case 3 subtracted from Case 6: Log plot of the velocity magnitude of the resultant time averaged velocity field between the velocity fields simulated with a $k-\epsilon$ turbulence model and a realizable $k-\epsilon$ turbulence model in OpenFOAM.

The average velocities of Cases 3 and 4 are plotted in Figures 4.15 and 4.14. The streamlines are plotted over the velocity fields indicate the direction of time averaged velocity fields. When plotting the difference between the average velocity fields of Case 3 and Case 6 as a logarithmic plot, as is illustrated by Figure 4.16, it is evident that in the case of two rows of cages simulated with a $k-\epsilon$ and realizable $k-\epsilon$ turbulence model, are identical for practical purposes, with a maximum deviation of approximately 0.2% of the incoming velocity.

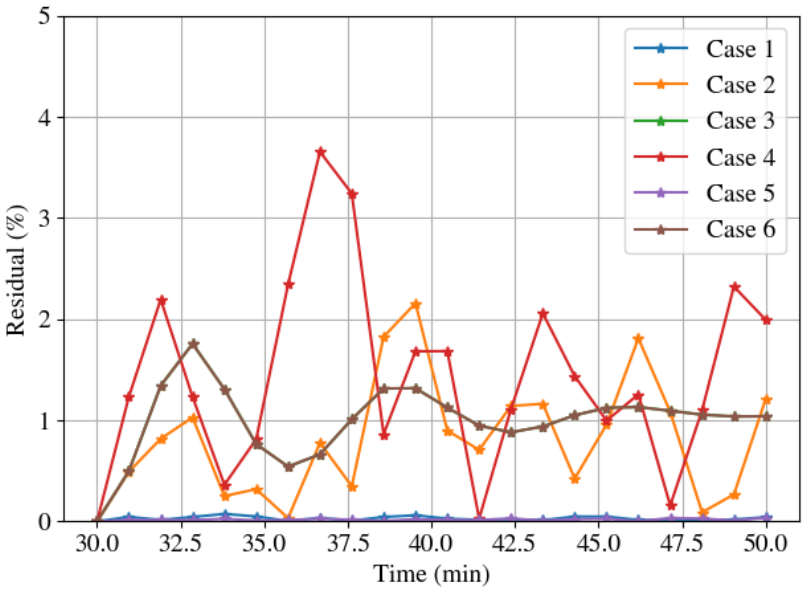


Figure 4.17: The average residuals at each timestep for the velocity field in all cases, as a percentage of the average velocity field.

The velocity residuals, as percentages, are plotted over time in Figure 4.17. This figure illustrates that the velocity field does not change substantially between timesteps for Cases 1, 3 and 5, indicating the there are no major fluctuations in the velocity fields. The velocity residuals for each case were calculated by the same means as those plotted in Figure 4.3. The residuals were calculated by finding the difference between the velocity fields, at each point in the grid, at sequential timesteps in each direction. This created two ΔU and ΔV velocity fields in the x and y directions respectively. The relative magnitude was then calculated by calculating the sum of the squares of the two directions, and dividing through by the velocity magnitude of the earlier timestep at the same position.

As expected, the residuals for the velocity fields of Cases 2 and 4 are not converging in Figure 4.17.

The result for Case 6, however, is unexpected, as the plot in Figure 4.16 indicates that the average velocity fields of Cases 3 and 6 were found to be almost identical. The residuals plot for Case 6 would be expected to converge around the same value as the plot for Case 3. However, the residual plot for Case 6 does appear to be converging around $0.0050\,\mathrm{ms}^{-1}$, which, as a relative velocity, is 1%, and acceptable as an indication of convergence. The maximum difference between timesteps is 3.65%, which is relatively low.

4.3 Velocity through the centreline

The velocity through the centreline, normalised with the inlet velocity can be compared to other simulations more accurately than the two-dimensional velocity field. The centreline velocity magnitude is plotted for each of the cases considered in the present study.

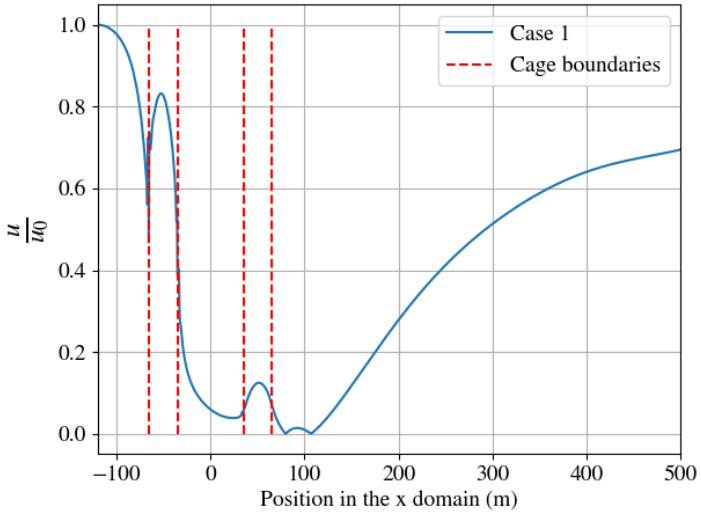


Figure 4.18: Case 1: one row with two cages with a $k-\epsilon$ turbulence model.

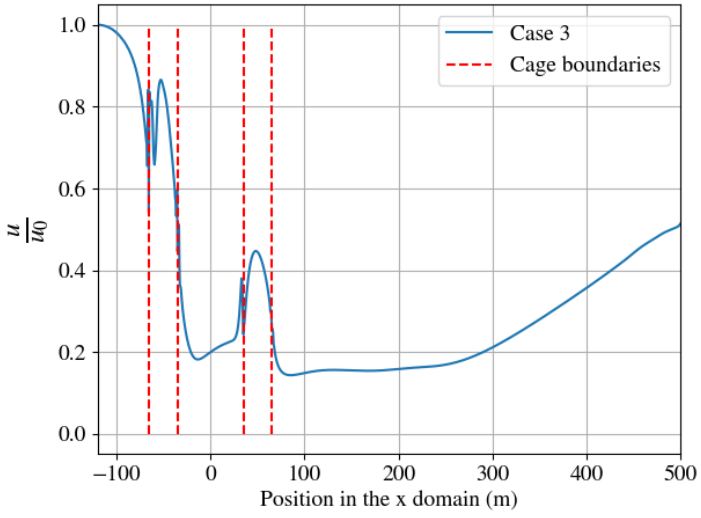


Figure 4.19: Case 3: two rows with two cages with a $k-\epsilon$ turbulence model.

As illustrated in Figure 4.18 (Case 1) and Figure 4.19 (Case 3), with a $k-\epsilon$ turbulence model in use, the velocity in the centre of the cages is approximately $0.8U_0$ where U_0 is the incoming velocity for both 1×2 and 2×2 cage formations.

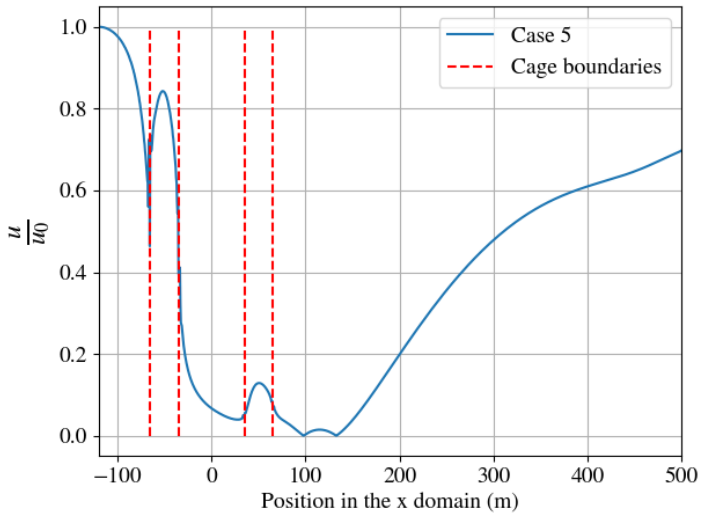


Figure 4.20: Case 5: one row with two cages with a realizable $k-\epsilon$ turbulence model.

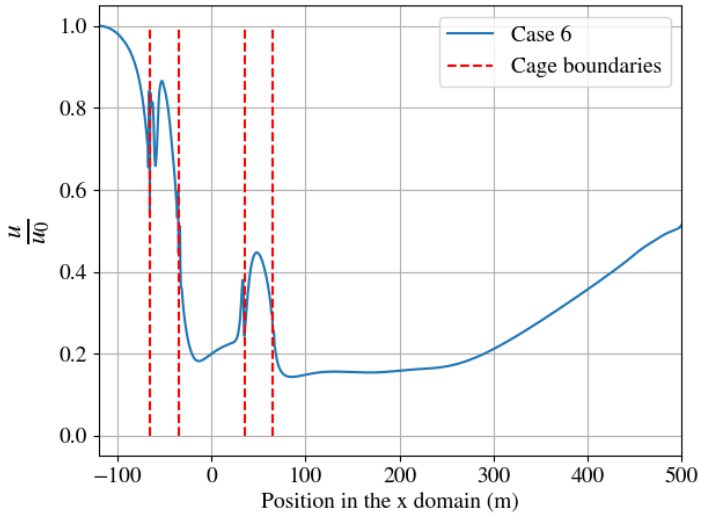


Figure 4.21: Case 6: two rows with two cages with a realizable $k-\epsilon$ turbulence model.

Figure 4.20 (Case 5) and Figure 4.21 (Case 6) illustrates the centreline velocity through the cages simulated with a realizable $k-\epsilon$ turbulence model.

By comparing Figure 4.18 (Case 1), and Figure 4.20 (Case 5) and comparing Figure 4.19 (Case 3) and Figure 4.21 (Case 6) the comparison between $k-\epsilon$ and realizable $k-\epsilon$ can be made. The relative differences are plotted as percentages in Figure 4.22 and confirm the difference that is shown in Figure 4.13 and Figure 4.16.

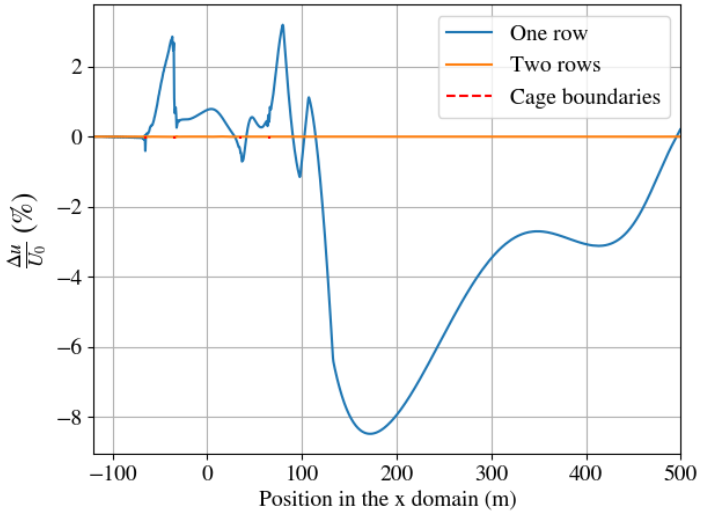


Figure 4.22: The change in velocity through the centreline between the simulations with $k-\epsilon$ and realizable $k-\epsilon$ turbulence models.

This shows that for practical purposes, a $k-\epsilon$ and realizable $k-\epsilon$ turbulence model result in similar velocity fields. The difference between realizable $k-\epsilon$ and $k-\epsilon$ is at most 12% of the incoming velocity. From here onwards, only Cases 1 to 4 will be considered, as the differences are minor.

As illustrated by Figure 4.23 (Case 2) and Figure 4.24 (Case 4) without turbulence modelling, the velocity through the centreline of the cages for both 1×2 and 2×2 cage formations are different.

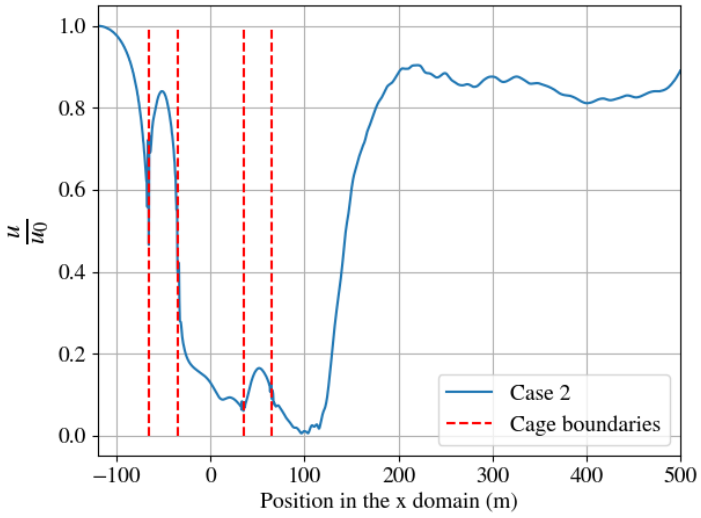


Figure 4.23: Case 2: one row with two cages with no turbulence model.

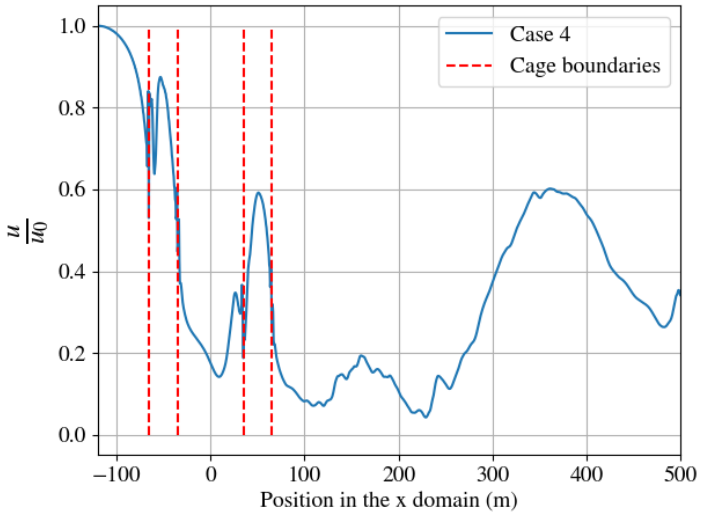


Figure 4.24: Case 4: two rows with two cages with no turbulence model.

Figures 4.18 to 4.24 indicate that different centreline velocities are observed with different turbulence models. The centreline velocities observed in Winthereig-Rasmussen et al. [2016], Bi and Xu [2018], Turner et al. [2015] as well as the present study, should be considered and compared.

In Winthereig-Rasmussen et al. [2016] the velocity through cages were simulated, and it was found when cages are in rows, the velocity through the centre

typically decreases for each cage. The study also plotted the difference in velocity through the centreline between one and two rows of cages and found that, in the case in which there are two rows, the velocity through the cages is higher than in the case in which there is a single row. By inspection of Figure 4.18 to Figure 4.21 this was also the case in the present study. Winthereig-Rasmussen et al. [2016] made use of the realizable $k - \epsilon$ turbulence model.

The computational study by Bi and Xu [2018], considering cages in two rows, showed a considerable reduction of velocity through the centre of the downstream cage. The velocity through the second cage in the row was 0.7 of the incoming velocity from the inlet boundary.

The experimental study conducted by Turner et al. [2015] did not plot the centreline velocities through the cages. Turner et al. [2015] does provide two-dimensional cross-sectional velocity fields that were obtained by the experimental equipment and that were interpolated. The orientation of the velocity plots obtained by Turner et al. [2015] are illustrated in Figure 4.25 by the shaded two-dimensional area. The orientation of the present study in comparison to the experimental study is illustrated in Figure 4.26 by the shaded two-dimensional area.

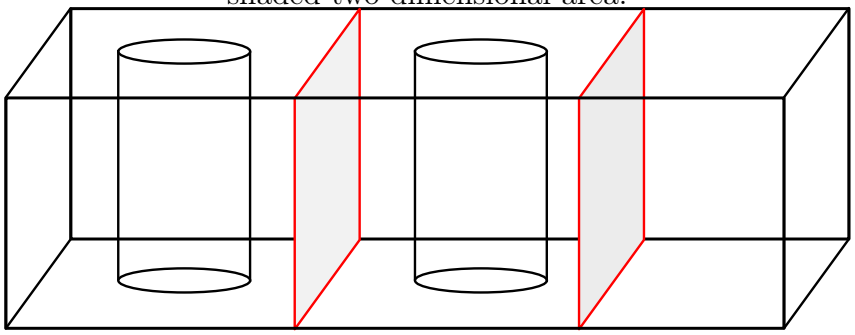


Figure 4.25: The orientation of the velocity fields with respect to the cages, obtained by Turner et al. [2015]. In the actual experiment, there is another cage, to keep this figure concise, the third cage was not added.

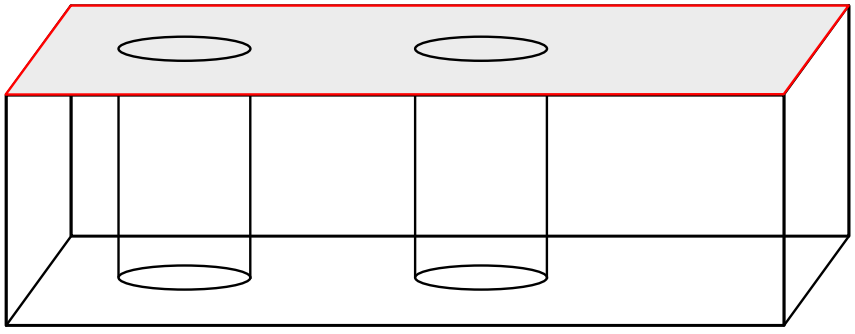


Figure 4.26: The orientation of the velocity fields relative to the cages of present study.

By comparing the velocities that overlap, as indicated in Figure 4.27, by the intersecting lines, in the plots in Turner et al. [2015] and the centreline plots in the present study, a comparison can be made.

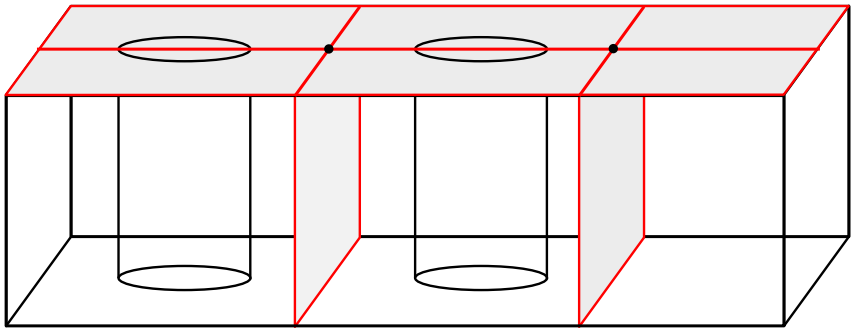


Figure 4.27: A visual aid for the velocities that should be compared in the present study and Turner et al. [2015].

In Turner et al. [2015], the velocity between the cages is greatly reduced, as is reflected in the centreline velocities from the present study, in Figure 4.18 to Figure 4.24. The studies by Winthereig-Rasmussen et al. [2016] and Bi and Xu [2018] did not observe a considerable reduction in velocity between farms.

Turner et al. [2015] considered two different set-ups, one where cages are in close proximity, $0.2\,d_c$, where d_c is the diameter of the cages, and one where cages are farther apart, $0.9\,d_c$. In the case where cages are in close proximity the velocity between the cages is seemingly more reduced than when the cages are farther apart.

The velocities obtained in the simulations in the present study, are similar to those found in Turner et al. [2015], Bi and Xu [2018] and Winthereig-Rasmussen et al. [2016], although the details of the motion is different, and the extent to which the velocity reduces varies from study to study. The general trend of velocity reduction appears to be similar. Chapter 6 will compare the results obtained by the particle tracking model and compare the final destinations of the particles after 1 h 30 min.

Chapter 5

In-house code: Fish Infection Simulation Helper (FISH)

The full model is discussed in this chapter as a summary of the integration of the processes discussed in Chapters 2, 3 and 4.

Figure 5.1 is a flow diagram of the Fish Infection Simulation Helper (FISH), which is discussed in this chapter. The FISH package contains four classes,

- `ParticleTracking`,
- `Population`,
- `Concentrations`,
- `Infections`.

These classes and their methods, instance creation, the attributes of those instances, and how the classes are used, are discussed in this chapter.

The fully developed, time-averaged velocity is used to simulate the movement of the virions by means of the FISH package. Note that the following description makes reference to the flowchart in Figure 5.1.

As discussed in Chapter 4, once the OpenFOAM velocity fields have been obtained, the OpenFOAM outputs are processed in Paraview. Paraview is an opensource CFD visualisation software, which can save the information at each cell and the position of each cell as a comma separated value (csv) file.

A `RunAll.py` script is implemented in FISH that allows the simulation to be run with one script. FISH contains a method that processes the Paraview csv outputs, this is the first method in the `RunAll.py` script. The velocity processing script interpolates the velocity fields spatially, such that a uniform grid is created at each time step. The interpolation method used is the

`scipy.interpolate.griddata` with a cubic interpolation scheme. These uniform grid velocity fields are then used to calculate the time averaged velocity field, which is saved as a NumPy array file (.npy).

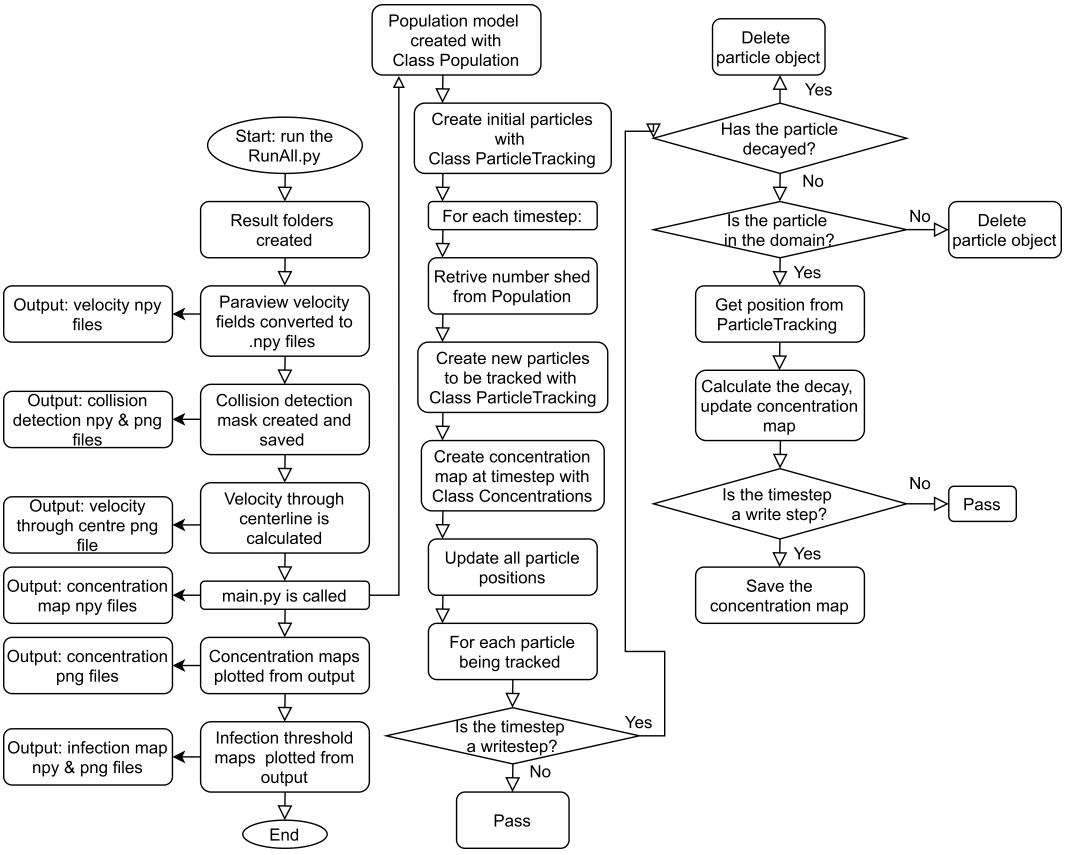


Figure 5.1: A flowchart describing the FISH package and how it is used.

A property set-up python dictionary script (a script that contains a hashmap) is implemented in FISH and allows the user to specify the parameters of the disease spread simulation. This hashmap is called into all the classes within FISH to ensure that the simulation is defined consistently. The script will be referred to by its filename, `property_setup.py`.

The script `property_setup.py` contains the parameters tabulated in Table 5.1. The parameter "Simulation" refers to the case in question and is a tag that indicates where the velocity outputs are stored. In this study the tags were related to the cases defined in Chapter 4, where Cases 1 to 6 were tagged as Sim1 to Sim6.

Table 5.1: The parameters in the property dictionary of the FISH package written in Python.

Parameter	**Value**	**Source**
Totaltime (s)	5400	-
Write step (s)	1800	-
Simulation	Sim1	-
Grid Size $[x, y]$ (m)	$[1240, 320]$	-
Grid boundary x	$[-120.0, 500.00]$	-
Grid boundary y	$[-80.00, 80.00]$	-
Time step, dt (s)	1	-
Diffusion coefficient, K_H (per s m^2)	1	[Salama and Murray, 2013]
Farm radius (m)	15	[Winthereig-Rasmussen et al., 2016]
Cluster size	5×10^{12}	-
Farm centres	$[[-50, 0], [50, 0], [-50, 35], [50, 35]]$	-
Decay	3.33×10^{-05}	[Salama and Murray, 2013]
Number of fish	9×10^4	[Watershed Watch Salmon Society, 2004]
Weight per fish (kg)	4.5	[Britannica, 2019]
Depth of cage (m)	13	[Winthereig-Rasmussen et al., 2016]
Infection threshold, ϕ (per ml kg)	0.10	[Salama and Murray, 2013]
Transmission rate, β (per s)	1.74×10^{-07}	[Salama and Murray, 2013]
Removal Rate, γ (per s)	4.63×10^{-07}	[Salama and Murray, 2013]
Infection rate, σ (per s)	1.62×10^{-06}	[Salama and Murray, 2013]
Shed rate (per s kg m^3)	200	[Salama and Murray, 2013]

The `property_setup.py` script also calculates other necessary properties to ensure that all properties are correct if any parameter is altered. These properties include the cage volume for a given radius, the weight of the fish in the cage, the shed rate of the cage per second kilogram meter squared, and the infection threshold per meter squared, as discussed in Section 3.8.

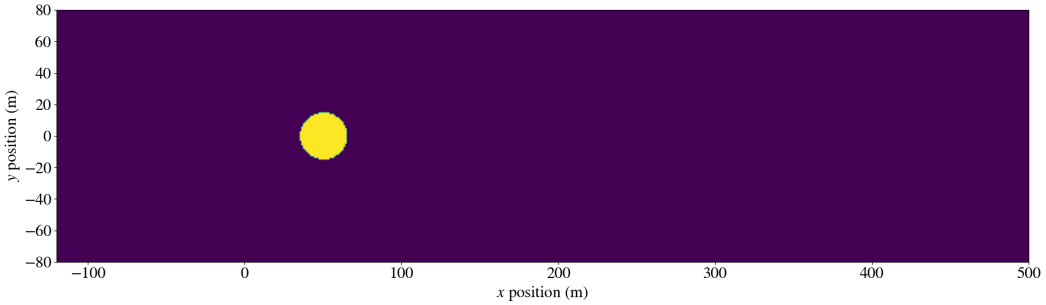


Figure 5.2: The collision detection mask used in Cases 1 and 2.

As part of the preprocessing, a collision detection mask is created. This is step three of the `RunAll.py` script. In this collision detection mask, all farms that are not infected initially are mapped with ones (1) in the farm and zeros (0) outside the farm. An example of the collision detection mask for Cases 1 and 2 is shown in Figure 5.2. The collision detection mask is then saved as a .npy file for further analysis.

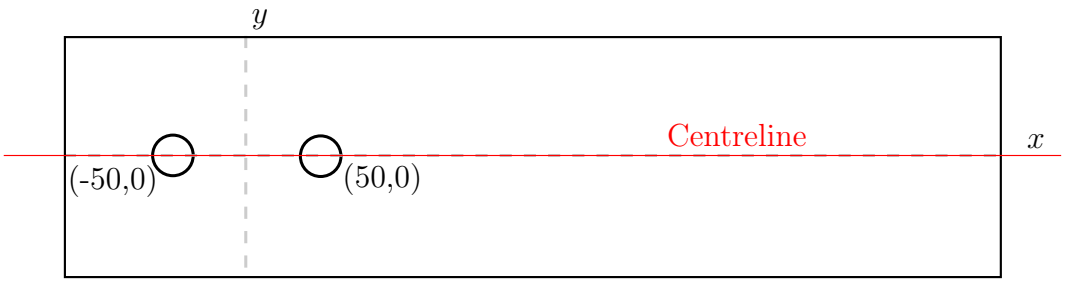


Figure 5.3: The description of how the centreline velocities are extracted in relation to the domain.

The fourth step of `RunAll.py` is to create the centreline plots. FISH outputs the centreline velocity plots, discussed in Section 3.9. The centreline velocity is calculated by extracting the velocities along the $x-$axis as is indicated in Figure 5.3 by the red line. The same position is used for an array of two rows with two cages in each. The centreline velocity is then plotted and saved as a .png image file.

The next step of `RunAll.py` is to call the main simulation script, `main.py`, the details of which will be discussed in Section 5.1.

Once the script `main.py` has been run, the script responsible for plotting the concentration maps is called, `concentration-maps.py`. This script imports each of the concentration map files, that are saved as .npy files, and plots the concentration maps and cages as two-dimensional plots and saves these plots as

.png image files. The concentration maps script ensures that the concentration maps at each write step are plotted and stored for further visual analysis.

The next script to be called in `RunAll.py` is the `infections-maps.py`, which is responsible for plotting the areas in which the infection is likely to spread. At each cell within the concentration map, the concentration of particles within the cell is compared to the infection threshold for the virus, as discussed in Section 3.9. The infection maps are calculated in a similar manner to the concentration maps in the `Concentrations` class, which will be discussed in Section 5.4. These infection maps are created and saved as .npy files and plotted in two dimensions, which are saved as .png image files for further visual analysis. The `Infections` class is described further in Section 5.5.

5.1 Main.py

The simulation script, `main.py`, is responsible for the coupling of the particle tracking simulation with the population model. Algorithm 1 works through the main.py.

The script makes use of the classes `Population`, `ParticleTracking` and `Concentrations` which are further discussed in Sections 5.2, 5.3 and 5.4.

Algorithm 1 The inner workings of `main.py`.

Import the properties hashmap file `property_setup.py`
S_0, E_0, I_0, R_0 are defined
A new population object instance is created,
`cage_population = Population(`S_0`,` E_0`,` I_0`,` R_0`)`
Create empty list to keep track of particle objects
`Particles = []`
New particle objects for *all* initial virus clusters within the boundary of the infected cage
`new_particle = ParticleTracking(`x `position,` y `position)`
`Particles.append(new_particle)`
for each timestep from 0 to total time in `property_setup.py` **do**
  Update the population model
  `cage_population.update_population()`
  Retrieve the number of particles shed at the timestep
  `clusters_shed = cage_population.get_shed()`
  for Each new particle cluster (from 0 to `clusters_shed`) **do**
    Create new particle objects, within the boundary of the infected cage, and append to the particle list
    `new_particle = ParticleTracking(`x `position,` y `position,` $t = 0$`)`
    `Particles.append(new_particle)`
  end for
  Create a new concentration map object
  `concentration_map = Concentrations(`x `size,` y `size)`
  for each particle in the `Particles` list **do**
    Update the particle's position
    `Particles[current particle].update_position()`
    if The wrtie timestep (in the hashmap) is a factor of the current timestep **then**
      if The particle has decayed **then**
        Delete the particle object, which removes it from the list
      end if
      if The particle is within the domain **then**
        Map the particle's effectiveness (with decay) to the concentration map at the particle's position
        `Particles[current particle].get_xposition()`
        `Particles[current particle].get_yposition()`
        `Concentrations.update_tally(`x `position,` y `position, decayed particle)`
      end if
    end if
  end for
  After iterating through all particles
  if The wrtie timestep (in the hashmap) is a factor of the current timestep **then**
    Save the concentration map as a NumpPy array file (.npy)
  end if
end for

5.2 Population class

The `Population` class makes use of the SEIR population model discussed in Section 2.2 and the shedding and decay model discussed in Section 3.8. The SEIR model used has the following equations to update each population,

$$S_{k+1} = S_k - \beta \frac{S_k I_k}{N_1} \Delta t, \tag{5.1}$$

$$E_{k+1} = E_k(1 - \sigma \Delta t) + \beta \frac{S_k I_k}{N_1} \Delta t. \tag{5.2}$$

$$I_{k+1} = I_k(1 - \gamma \Delta t) + \sigma E_k \Delta t, \tag{5.3}$$

$$R_{k+1} = R_k + \gamma I_k \Delta t. \tag{5.4}$$

The terms in each equation can be broken down as follows, S_k, E_k, I_k and R_k, are the terms that refer to the size of the population in the susceptible, exposed, infected and recovered populations, respectively at the current timestep k. The terms β, σ, γ and Δt are the transmission rate, infection rate, removal rate and time increment respectively. This model is discussed at length in Chapter 2.

Once the populations are calculated for the timestep, the size of the infected population is used to calculate the number of new virus particles shed,

$$\mathcal{W}_{k+1} = \Gamma I_k, \tag{5.5}$$

where $\mathcal{W}$ is the waterphase (number of particles travelling between the cages), Γ is the shedding rate and I_k is the number of infected fish at the previous timestep k, as discussed in Section 3.8. The decay of the virus particle is not calculated in the `Population` class, this is calculated in the `main.py` script.

The class constructor creates an object that contains the susceptible population, exposed population, infection population and recovered population. Only the two most recent values for each population are stored in a list to improve the space complexity of the algorithm.

The creation of an instance of this class is shown in Algorithm 1. The `Population` creator is given the initial population values S_0, E_0, I_0, R_0 and creates an object that keeps track of each population value at the current timestep.

The `Population` class contains the following methods, as listed below.

- `update_population()`: This method is called on the object and updates each population according to equations (5.1) to (5.4). When the new populations are calculated, the list is shifted left such that the oldest populations are removed, and the newest populations are appended to the list. This ensures that the list is always of length two. There is currently no method to retrieve the list for each population, only the most recent position.

- `get_shed()`: Returns the number of clusters shed using the most recent infection population, by equation (5.5).
- `get_S()`: Returns the susceptible population at the current timestep.
- `get_E()`: Returns the exposed population at the current timestep.
- `get_I()`: Returns the infected population at the current timestep.
- `get_R()`: Returns the recovered population at the current timestep.

5.3 ParticleTracking class

The `ParticleTracking` class makes use of a two-dimensional Lagrangian particle tracking model to simulate the path of each particle object in two dimensions. In Section 3.3, mathematical models of various particle tracking algorithms are discussed. The particle tracking algorithm used in this class is derived as follows. This model begins with the two Lagrangian equations of motion in two directions,

$$\frac{dx_p}{dt} = u_p(x, y), \tag{5.6}$$

$$\frac{dy_p}{dt} = v_p(x, y), \tag{5.7}$$

where x_p, y_p are the position and u_p, v_p are the velocities of the particle p, and t is the time. The Lagrangian equations of motion can be approximated with Euler's method as follows,

$$x_p^{k+1} = u_p(x, y)\Delta t + x_p^k \tag{5.8}$$

$$y_p^{k+1} = v_p(x, y)\Delta t + y_p^k \tag{5.9}$$

where x_p and y_p are the x and y positions of the current particle p, k is the current timestep, and Δt is the time increment.

The velocities u_p and v_p at the particle's position are bilinearly interpolated in space,

$$\begin{aligned} u_p^k =& m\,n\,u_{(i,j)}^k + (1-m)\,n\,u_{(i+1,j)}^k \\ &+ (1-m)\,(1-n)\,u_{(i+1,j+1)}^k + m\,(1-n)\,u_{(i,j+1)}^k \end{aligned} \tag{5.10}$$

$$\begin{aligned} v_p^k =& m\,n\,v_{(i,j)}^k + (1-m)\,n\,v_{(i+1,j)}^k \\ &+ (1-m)\,(1-n)\,v_{(i+1,j+1)}^k + m\,(1-n)\,v_{(i,j+1)}^k, \end{aligned} \tag{5.11}$$

where m and n are the fractional position of the particle p within the cell as in Figure 3.5.

The diffusion term is then added to the particle's position, as is discussed in Section 3.6.

The final x_p position is then obtained by combining equations (5.8) and (5.10) and adding the diffusion term in the x direction,

$$x_p^{k+1} = x_p^k + u_p^k \Delta t + \mathcal{Z}_1 \sqrt{\Delta t \, K_H}. \tag{5.12}$$

The equivalent final y_p position combines equations (5.9) and (5.11) with the diffusion in the y direction,

$$y_p^{k+1} = y_p^k + v_p^k \Delta t + \mathcal{Z}_2 \sqrt{\Delta t \, K_H}, \tag{5.13}$$

where $\mathcal{Z}_1$ and $\mathcal{Z}_2$ are random numbers between 0 and 1, these two values are calculated as a vector of independent random numbers that have a standard normal distribution. The term K_H is the horizontal diffusion, the value of which is in Table 5.1.

The class `ParticleTracking` constructs an object that has an x position, a y position and a time t, which is the time that the particle has been in the simulation. The time attribute is used to determine by how much the particle has decayed at the current timestep.

The creation of an instance of this class is shown in Algorithm 1. The `ParticleTracing` creator is given the x and y positions at which a new particle is created and a time which should be set to $t = 0$ initially. As in the class `Population`, only the two most recent (previous and current) positions of the particle are stored for both the x and y positions.

The methods in `ParticleTracing` are listed below.

- `update_position()`: This method is called on a particle object, and updates the time attribute by adding the time increment, and then updates the particle position by making use of equations (5.10) to (5.13). The calculation of the random term in Python is discussed in Section 3.5. When the new particle location is calculated, the list is shifted left such that the oldest particle location is removed, and the newest location is appended to the list. This ensures that the list is always of length two (2). There is currently no method to retrieve the position list. Only the most recent position.
- `get_xposition()`: This returns the most recent x position.
- `get_yposition()`: This returns the most recent y position.
- `get_time()`: This returns the time, in the units of the time increment, that the particle has been in the simulation.

5.4 Concentrations class

The `Concentrations` class creates a concentration map object which is a grid of zeros of a given size. The size of the concentration map is chosen depending on the resolution required by the simulation. For this study the concentration maps are created such that the cells represent $1\,\mathrm{m}^2$.

The methods in `Concentrations` are listed below.

- `update_tally()`: The method updates the concentration map. The method is given the particle's x and y positions, and portion of the cluster that is still effective, i.e. if enough time has passed such that only half of the cluster has not decayed, 0.5 is passed into the `update_tally()` function.
- `get_concentrations()`: Returns the concentration map at the current timestep.

5.5 Infections class

The `Infections` class is similar to the `Concentrations` class. The `Infections` class creates an infection map. The class constructor requires the current concentration map, and an x and y size of the map, to create a grid of zeros of the given size. This allows the infection maps to have a lower resolution than the concentration maps. A higher resolution is not possible as there is no way to retrieve the particle locations from the concentration maps.

- infection_map(): for each x, y position passed into the function, the concentration at that point is considered, if the concentration is above the infection threshold, the value in the grid is set to one (1), or else the value in the grid remains zero (0). This method should be called for the position of each concentration map cell to populate the infection map.
- get_infection_threshold(): Returns the infection map at the current timestep.

Chapter 6

Simulation results

In this chapter the results of various simulation runs for the evaluation of the particle tracking model are discussed. In addition, the results using different turbulence models were investigated. The various OpenFOAM velocity field outputs are used along with the FISH package to simulate the spread of the virus particles throughout the domain. The concentration and infection plots are discussed in this chapter.

6.1 Case 1 and 2: particle tracking on the velocity fields with one row of cages

The particle tracking simulation was run for Case 1 and Case 2, as described in Section 4.1, where the set-up consists of a single row of two cages in each row, 1×2.

In Figure 6.1 the distribution of the virions can be seen at 1 h 30 min, which illustrates the motion of the virus clusters over the timespan. The virions tend to cluster between the two cages as the velocity is reduced considerably in this region. After 1 h 30 min the particles had not completely made it through the centre of the cage, as can be seen by the section that is void of particles behind the second cage. This is also the case in Figure 6.2 although the tail of particles behind the cages is slimmer in Case 2.

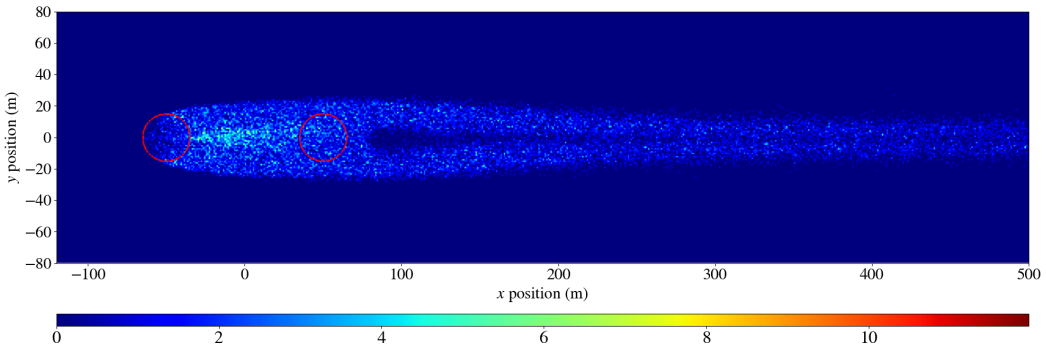


Figure 6.1: Case 1, with $k - \epsilon$ turbulence model: Concentration dispersion of virus clusters after 1 h 30 min.

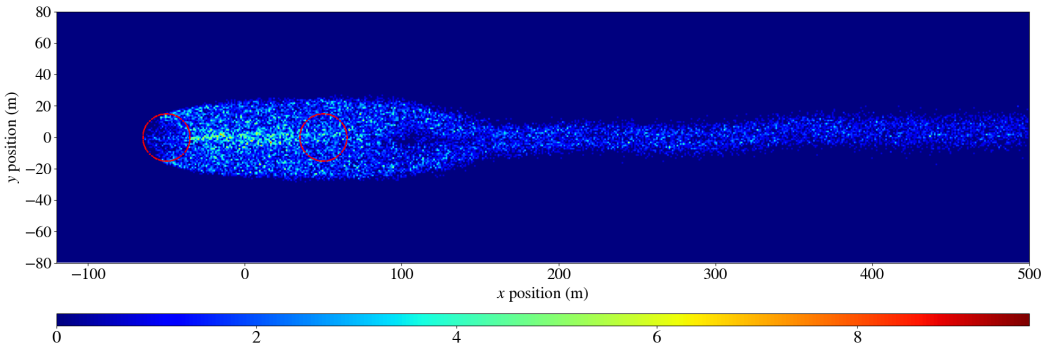


Figure 6.2: Case 2, with no turbulence model: Concentration dispersion of virus clusters after 1 h 30 min.

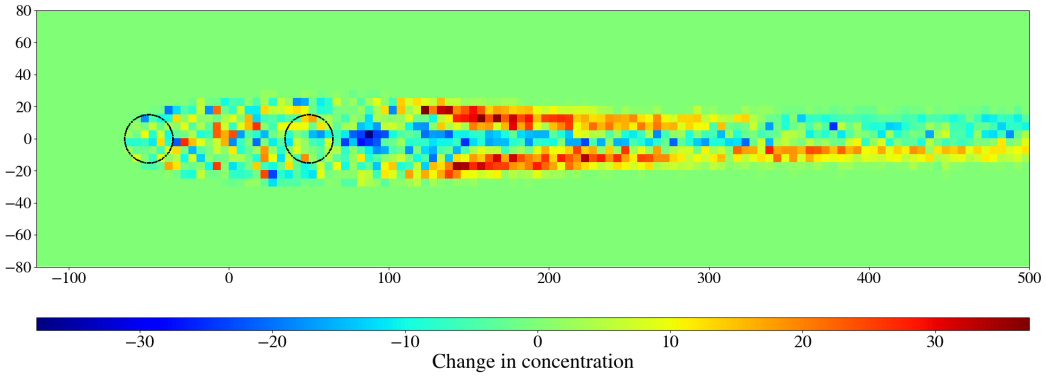


Figure 6.3: Concentration plot of Case 2 subtracted from the concentration plot of Case 1 to illustrate the difference in dispersion patterns created by the two models.

The difference between the two concentration distributions is shown in Figure 6.3. The grid of the map in Figure 6.3 is coarser than that of Figures 6.1 and 6.2, as the average of the concentration field is shown more clearly with a coarser grid. This figure confirms the difference in tail width between Case 1 and Case 2, and confirms that the infection distribution through the centreline of both cases are similar.

The collision detection mask, discussed in Chapter 5, for Cases 1 and 2 is used to count the number of virion clusters in the cages in Table 6.1 for five separate runs of the particle tracking simulation. The average difference in number of particles that reach the downstream cages is tabulated in Table 6.1

Table 6.1: The number of virus clusters in the initially uninfected farms at 1 h 30 min.

Time	Case	Average number of particles	Average difference (%)
0 h 30 min	1	88.0	18.4
	2	107.9	
1 h 00 min	1	614.6	2.2
	2	628.2	
1 h 30 min	1	1171.7	-1.5
	2	1154.9	

As can be seen by the average number of clusters, detected by the collision detection mask in the downstream cages, the results are similar in Cases 1 and 2. There is an average absolute difference of 7% in the number of particles in the downstream cages between Cases 1 and 2.

The infection maps, discussed is Chapter 5, are created for the concentration plots of Case 1 and 2. Figures 6.4 and 6.5 highlight the high risk areas where each plotted point is above the infectious threshold. This makes use of the smallest minimum infectious threshold as calculated in Section 3.9 which is compared to the concentration in each cell. Upon investigation of Figures 6.1 to 6.5, it is evident that within even the least concentrated areas, the fish are still at risk of contracting the disease.

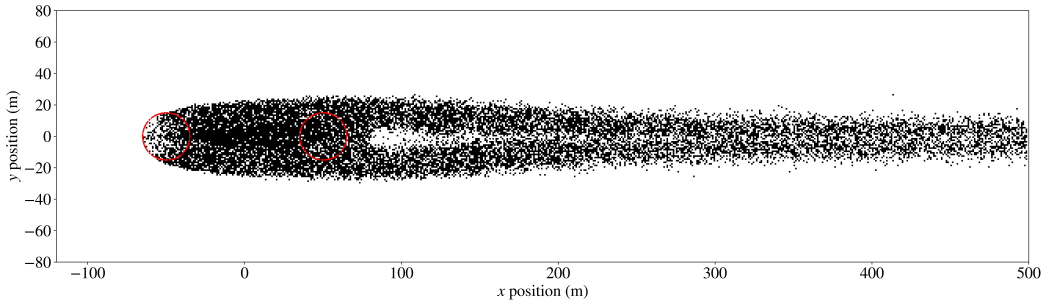


Figure 6.4: Case 1, with $k - \epsilon$ turbulence model: Dispersion of the high risk areas after 1 h 30 min.

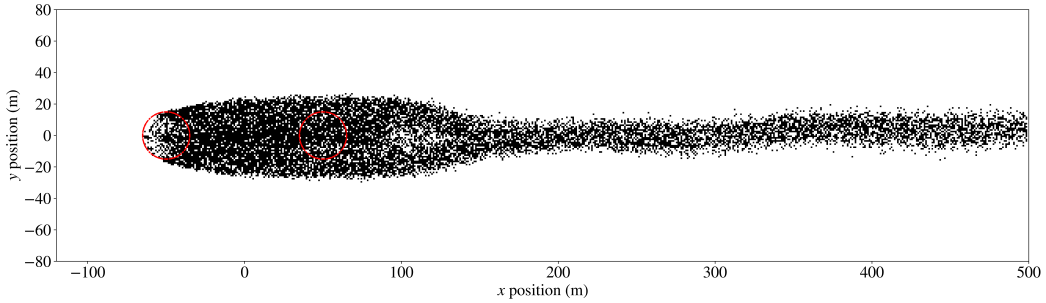


Figure 6.5: Case 2, with no turbulence model: Dispersion of the high risk areas after 1 h 30 min.

6.2 Case 3 and 4: particle tracking on the velocity fields with two rows of cages

The particle tracking simulation was then repeated for Case 3 and Case 4 where the set-up consists of two rows of two cages in each row, 2×2, as is shown in Figure 4.2.

As is illustrated by Figures 6.6 and 6.7, far fewer virus clusters reach the downstream cage than in the case of a single row of cages. Instead these particles pass by the cage to the right, relative to the flow orientation. This can be expected by considering the streamline plots of Cases 1 and 2 in Figures 4.11 and 4.10 respectively, as well as the streamline plots of Cases 3 and 4 in Figures 4.14 and 4.15 respectively. As the particles follow the flow, and thus the streamlines, the difference in particle distribution between the cage orientations is expected. The streamlines of the average velocity fields through Cases 1 and 2 indicate that the particles will be distributed across the downstream cages evenly, which was seen in Figures 6.1 and 6.2. The velocity field streamlines of Cases 3 and 4, indicate that the particles in the downstream cage will have a higher concentration towards the lower boundary of the downstream cage, which is observed in Figures 6.6 and 6.7.

Comparing the results shown in Figure 6.6 and Figure 6.7, it is evident that due to the vortex shedding, the trajectory of the virus clusters further downstream, behind the two final cages, is different in Cases 3 and 4.

The difference between the two concentration distributions of Cases 3 and 4 is shown in Figure 6.8. The concentration distribution in Figure 6.8 through the centre of the cages and behind the cage is similar, with the exception of the area in which the vortices shed have an influence on the average velocity. The results seen in Figure 6.8 are similar to those observed in section 6.1.

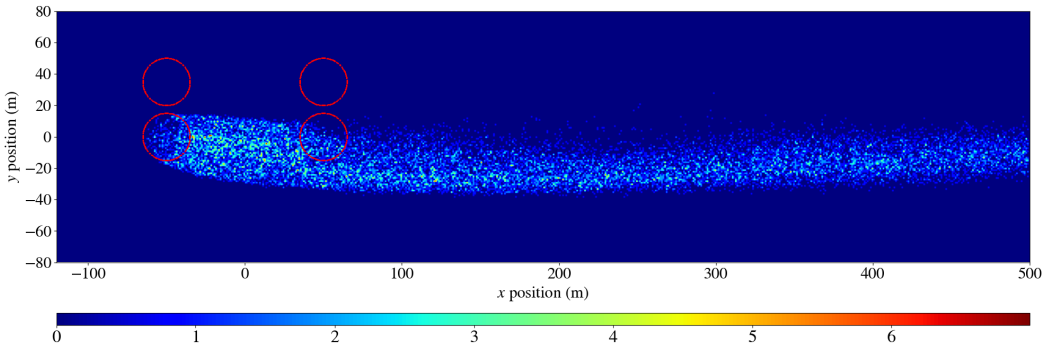


Figure 6.6: Case 3, with $k - \epsilon$ turbulence model: Concentration dispersion of virus clusters after 1 h 30 min.

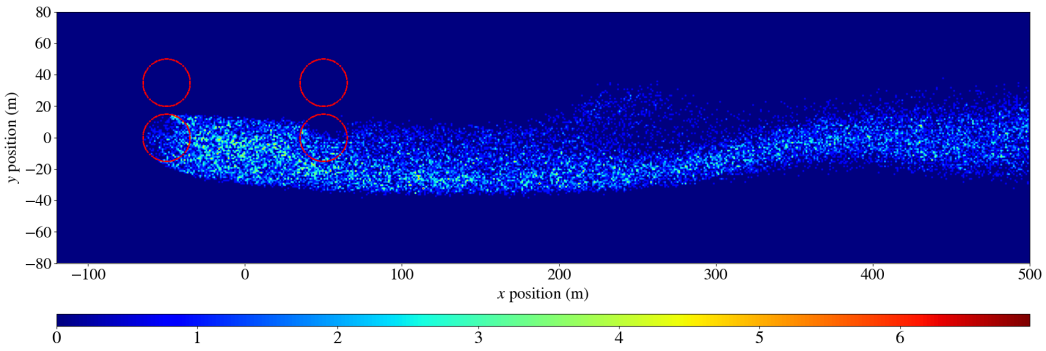


Figure 6.7: Case 4, with no turbulence model: Concentration dispersion of virus clusters after 1 h 30 min.

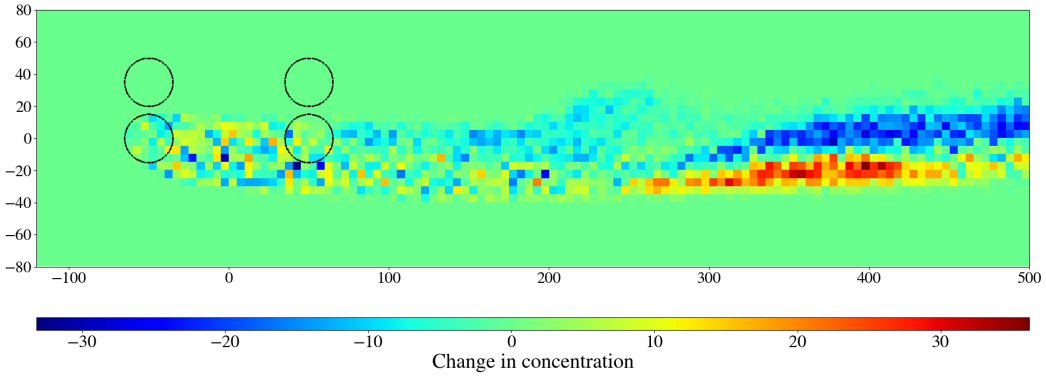


Figure 6.8: Concentration plot of Case 4 subtracted from the concentration plot of Case 3 to illustrate the difference in dispersion patterns created by the two models.

The collision detection mask for two rows of cages is used to calculate the number of virus clusters that reach the downstream cages for both Case 3 and Case 4. Comparing the average number of clusters in the downstream cages, at various timesteps, Table 6.2 describes the similarity between the two cases, with an average absolute difference of 5.3% particles in the downstream cages. This indicates that for one or two rows of cages, both the simulation without a turbulence model and that with a $k-\epsilon$ turbulence model, and by extension a realizable $k-\epsilon$ model, yield a similar result for the spread of a virus within the set-ups simulated in this study.

Table 6.2: The number of particles in the uninfected farms

Time	**Case**	**Average number of particles**	**Average difference (%)**
0 h 30 min	3	213.5	-5.8
	4	201.7	
1 h 00 min	3	487.6	6.7
	4	522.8	
1 h 30 min	3	817.6	3.3
	4	845.5	

Similar to the results illustrated in Figures 6.4 and 6.5, these results, shown in Figures 6.9 and 6.10 illustrate that the fish are at risk of contracting the disease even if the concentration is relatively low in that cell.

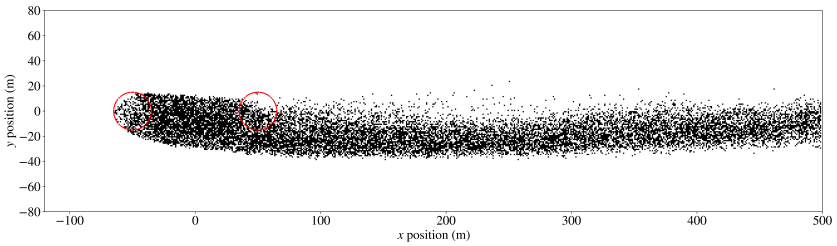


Figure 6.9: Case 3, with $k - \epsilon$ turbulence model: Dispersion of the high risk areas after 1 h 30 min, where the area is either infectious (black) or non infectious (white).

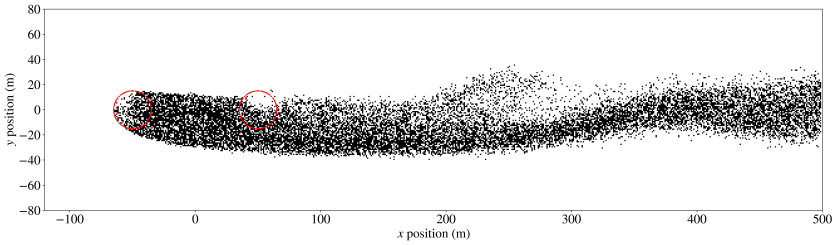


Figure 6.10: Case 4, without turbulence model: Dispersion of the high risk areas after 1 h 30 min, where the area is either infectious (black) or non infectious (white).

The results in this chapter are discussed further in Chapter 7 and conclusions are drawn from the simulation results of this study.

Chapter 7

Conclusions

7.1 Summary

This study set out to model, by means of computational fluid dynamics, the flow through and past an array of salmon fish cages, which is then coupled to Lagrangian models for the prediction of the spread of water borne infectious diseases between the cages within a farm.

In Chapter 2 various disease models were compared for different disease mechanics. The susceptible infected and recovered (SIR) and susceptible exposed infected abd recovered (SEIR) models were compared in terms of modelling infectious salmon anaemia virus (ISAV). The SIR model is preferred for diseases that do not have incubation periods and where the infection is transmitted to an uninfected fish only if the host is infected and showing symptoms. The SEIR model was developed to simulate disease spread for diseases where the disease has an incubation period and can be transmitted while the host is asymptomatic, as is the case with ISAV. The initial values for the disease population model were chosen based on the worst case scenario, ensuring that the maximum number of fish possible were infected. In this population model, the population of interest is the infected population of fish. The number of infected fish at a given timestep was used to calculate the number of new particles in circulation at each timestep.

In Chapter 3, the particle tracking algorithms and all relevant parameters are discussed. An optimal size for a cluster of virions was determined to be 1×10^{13} virions. For the simulations in this study 5×10^{12} virions per cluster were used, as it was found that this cluster size allowed for the best analysis of the simulation results. This value is limited in that no guidelines were found during the course of the present study to indicate how to choose an optimal value for the cluster size. The value was chosen through trial and error, future work should relook at this value and re-evaluate the chosen cluster size in this study, and the effect it has on the dispersion of virus particles. The particle

tracking algorithms were then discussed, MODPATH [Pollock, 2016], TRACE [Beletsky et al., 2007], SINCEM [Fabbroni, 2009] and SINCEM2 [Fabbroni, 2009]. The ARIANE algorithm [Blanke and Raynaud, 1997] was also discussed in Appendix C. The particle tracking accounts for the advection term of the motion of the particle clusters.

The MODPATH algorithm developed for MODFLOW, calculated the point at which a particle exited the cell. This made it complicated to add a diffusive term as the timestep between particles entering and exiting cells would not be constant. The MODPATH algorithm was also not used as it was computationally expensive (as it required the calculation of the time taken to exit over a cell face for each dimension, and the calculations would continue for the shortest exit time).

The TRACE algorithm made use of velocities and their derivatives which are bilinearly interpolated temporally and the particle position is second order in time. The solution to this equation is discussed in Appendix B.

The SINCEM2 algorithm made use of fourth order Runge-Kutta for the interpolation of the velocities in time. As this study considers a stationary velocity field, this is not necessary.

The SINCEM algorithm made use of a velocity term that is linearly interpolated in time and bilinearly interpolated temporally. This leads to a simplistic yet effective solution. The final particle tracking model was based on the approach taken to construct the SINCEM model and is discussed in Chapter 5.

The single particle's motion is made up of an advection and diffusion term, as is discussed in Chapter 3. The diffusion term was based on a model for Brownian motion which makes use of Langevin equations and a Weiner process. This makes up for the diffusive term in the particles motion, as discussed in Section 3.5. The shedding of the virions is modelled with a linear equation where the number of particles shed at each timestep is directly proportional to the number of infected individuals at the timestep. The virions experience an exponential decay that is a function of time, with a constant decay coefficient. Both the decay and shedding are used to keep track of the number of virion clusters that are in circulation, as discussed in Section 3.8.

The disease is likely to be transmitted only if the particle concentration is greater than a given minimum infection threshold. This infection threshold was used to highlight the areas in the domain where fish are at high risk of contracting the disease, as discussed in Section 3.9. In the case of infectious salmon anaemia virus, the minimum infection threshold is low, and thus the disease is highly infectious and even a small amount of virions was likely to infect the downstream cages.

The particle tracking algorithm chosen made use of temporally bilinearly interpolated velocities and a particle position that is a first order time. New

particles were created and tracked, based on the shedding and the time a virion cluster remains in the simulation was dependent on the decay. The final model used in the simulations was discussed in Chapter 5.

In Chapter 4 the extent to which the cages influenced the velocity field and the extent to which the turbulence model altered the velocity field were discussed. One simulation with no turbulence model and two simulations with turbulence models were considered on one row of two cages and two rows of two cages. The two turbulence cases included a $k-\epsilon$ turbulence model and a realizable $k-\epsilon$ turbulence model. In the case of one row of two cages, the $k-\epsilon$ turbulence model and realizable $k-\epsilon$ turbulence model was found to deviate slightly with a maximum difference of approximately 11.3%. In the case of two rows of two cages, the $k-\epsilon$ turbulence model and realizable $k-\epsilon$ turbulence models yielded the same results. Thus the turbulence model that was used was the $k-\epsilon$ turbulence model and the realizable $k-\epsilon$ turbulence model was not used as a velocity field in the particle tracking simulation.

In Chapter 5 the particle tracking algorithm was coupled with a population model in an in-house Python package called Fish Infection Simulation Helper (FISH). The code made use of a SEIR population model and a particle tracking model based on the SINCEM2 algorithm. FISH was used in post-processing to simulate the spread of waterborne diseases within the domain on the OpenFOAM velocity fields to obtain the results of this study.

In Chapter 6 the outputs for each case of velocity field was compared by means of simulating the spread of virions from an infected farm to the farms downstream, by means of the FISH package in Chapter 5. The results of the particle tracking were then used to evaluate the spread of disease as well as the difference the turbulence model made to overall simulation. It was found that when comparing the $k-\epsilon$ turbulence model and no turbulence model, the results were similar. This suggests that the simulation of flow through and around an array of fish cages in a fjord, could be simulated with a $k-\epsilon$ turbulence model or realizable $k-\epsilon$ turbulence model to obtain a uniform velocity field, as the time averaged velocity field yielded the same results over time. This implies that, for similar conditions, the velocity fields can be simulated without a turbulence model, or with a $k-\epsilon$ or realisable $k-\epsilon$ turbulence model, and similar results will be observed. These results are however limited to the systems with the Reynolds numbers stated in the study.

In Chapter 6 it was also found that farms with a two row layout experienced less infection spread than farms with a single row. This confirms that the industry standard, where cages are in at least two rows, could lead to less disease spread within a single farm. These results are limited as the only one orientation of flow through the cages. Staggered cages should also be considered as it will impact the transmission of disease and the extent to which the staggering plays

a roll.

7.2 Future work

There are numerous cases which should still be considered in order to obtain a comprehensive analysis of the flow fields through and around an array of porous cylinders, and thus the effect of these flow fields of the spreading of infectious diseases.

The arrangement of various cage staggering patterns should also be investigated. The only case considered in this study is where the cage rows are aligned with the incoming constant velocity, which, as was seen in previous studies, is the worst case scenario for velocity reduction.

Larger arrays of cages at varying distances should also be investigated. Further work should consider varying magnitudes of incoming velocities to evaluate the flow patterns and there effect on the spreading of virions. The addition of a tidal component can also be added as a user defined function in OpenFOAM. In the fjords in which these cages are placed, there is often a tidal component to the velocity field and it should be considered for a comprehensive model.

Scanlon [2019] discusses `marineFoam`, a CFD tool that aims to model the flow through and around fish cages in open water, similar to the cages considered in this study. Scanlon [2019] makes use of the $k - \epsilon$ turbulence model, similar to this study. The solver `marineFoam` may also be used in future work and should be compared to the solver used in this study.

The work done in this study along with the future work mentioned in this chapter, can be used to find the optimal placement of salmon cages within a farm in a fjord to minimise disease spread.

Bibliography

Aerospace Engineering (2016). The von kármán vortex street and Tacoma Narrows disaster. https://aerospaceengineeringblog.com/the-von-karman-vortex-street-and-tacoma-narrows-disaster/. Accessed: 05 June 2020.

Allen, J. (2001). Salmon farm cages below Toravaig. https://www.geograph.org.uk/photo/2567640. Online: accessed 08 April 2020.

Allen, L. J. S. (1994). Some discrete-time SI, SIR, and SIS epidemic models. *Mathematical Biosciences*, 124(1):83 – 105.

Beletsky, D., Mason, D. M., Schwab, D. J., Rutherford, E. S., Janssen, J., Clapp, D. F., and Dettmers, J. M. (2007). Biophysical model of larval yellow perch advection and settlement in lake michigan. *Journal of Great Lakes Research*, 33(4):842 – 866.

Bennett, J. and Hutchinson Clites, A. (1987). Accuracy of trajectory calculation in a finite-difference circulation model. *Journal of Computational Physics*, 68(2):272 – 282.

Bi, C. W. and Xu, T. J. (2018). Numerical study on the flow field around a fish farm in tidal current. *Turkish Journal of Fisheries and Aquatic Sciences*, 18:705 – 716.

Blanke, B. and Raynaud, S. (1997). Kinematics of the Pacific Equatorial Undercurrent: An eulerian and lagrangian approach from gcm results. *Journal of Physical Oceanography*, 27(6):1038 – 1053.

Bricknell, I. (2017). Chapter 3 - types of pathogens in fish, waterborne diseases. In Jeney, G., editor, *Fish Diseases*, pages 53 – 80. Academic Press.

Britannica (2019). Salmon. https://www.britannica.com/animal/salmon. Accessed: 16 May 2019.

Canadian Food Inspection Agency (CFIA) (2020). Species directory: Atlantic salmon - protected. https://www.inspection.gc.ca/animal-health/aquatic-animals/diseases/reportable-diseases/isa/fact-sheet/eng/1327198930863/1327199219511. Accessed: 20 February 2020.

Chen, H. and Christensen, E. D. (2016). Investigations on the porous resistance coefficients for fishing net structures. *Journal of Fluids and Structures*, 65:76 – 107.

Dagestad, K., Röhrs, J., Breivik, O., and Adlandsvik, B. (2018). Opendrift v1.0: A generic framework for trajectory modelling. *Geoscientific Model Development*, 11:1405 – 1420.

Fabbroni, N. (2009). Numerical simulations of passive tracers dispersion in the sea. Master's thesis, University of Bologna.

Fish Pool Index (2017). Monthly and annual average. `http://fishpool.eu/price-information/spot-prices/history/`. Accessed: 20 February 2020.

Gardiner, C. W. (1983). *Handbook of Stochastic Methods: for Physics, Chemistry and the Natural Sciences*. Springer Berlin Heidelberg.

Global Salmon Initiative (2020a). Sustainability report. `https://globalsalmoninitiative.org/en/sustainability-report/protein-production-facts/#edible-yield`. Accessed: 07 February 2020.

Global Salmon Initiative (2020b). What is salmon farming and why do we need it? `https://globalsalmoninitiative.org/en/what-is-the-gsi/what-is-salmon-farming-and-why-do-we-need-it/`. Accessed: 07 February 2020.

Gregory, A., Munro, L., Snow, M., Urquhart, K., Murray, A., and Raynard, R. (2009). An experimental investigation on aspects of infectious salmon anaemia virus (ISAV) infection dynamics in seawater atlantic salmon, Salmo salar L. *Journal of fish diseases*, 32:481 – 489.

Howard, B. (2014). Salmon farming gets leaner and greener. `https://www.nationalgeographic.com/news/2014/3/140319-salmon-farming-sustainable-aquaculture/`. Accessed: 19 February 2020.

Institute for Disease Modeling (2019). SEIR and SEIRS models. `https://idmod.org/docs/general/model-seir.html`. Accessed: 2018-04-05.

Jamieson, N. D. (2007). Infectious salmon anemia virus (isav). `http://homepage.usask.ca/~vim458/virology/studpages2007/Nicole/isav.html`. Accessed: 15 August 2018.

Kar, D. (2016). Chapter 1 - introduction. In Kar, D., editor, *Epizootic Ulcerative Fish Disease Syndrome*, pages 1 – 19. Academic Press.

Kasper, R. (2017). Particle simulation with openfoam: Introduction, fundamentals and applications. Accessed: 13 January 2020.

Kibenge, F. and Kibenge, M. (2016). Chapter 19 - orthomyxoviruses of fish. In Kibenge, F. S. B. and Godoy, M. G., editors, *Aquaculture Virology*, pages 299 – 326. Academic Press, San Diego.

Klebert, P., Patursson, Ø., Endresen, P. C., Rundrop, P., Birkevold, J., and Winthereig-Rasmussen, H. (2015). Three-dimensional deformation of a large circular flexible sea cage in high currents: Field experiment and modeling. *Ocean Engineering*, 104:511 – 520.

Li, M. Y., Graef, J. R., Wang, L., and Karsai, J. (1999). Global dynamics of a SEIR model with varying total population size. *Mathematical Biosciences*, 160(2):191 – 213.

Live Science Staff (2009). Milestone: 50 percent of fish are now farmed. `https://www.livescience.com/5682-milestone-50-percent-fish-farmed.html`. Accessed: 19 February 2020.

Microbiology Society (2016). Are viruses alive? `https://microbiologysociety.org/publication/past-issues/what-is-life/article/are-viruses-alive-what-is-life.html`. Accessed: 05 June 2020.

Milliken, E. (2017). The probability of extinction of infectious salmon anemia virus in one and two patches. *Bulletin of Mathematical Biology*, 79(12):2887 – 2904.

Milliken, E. and Pilyugin, S. (2016). A model of infectious salmon anemia virus with viral diffusion between wild and farmed patches. *Discrete and Continuous Dynamical Systems - Series B*, 21:1869 – 1893.

Mosdell, S. (2007). Ruakaka bay salmon farm. `https://www.flickr.com/photos/sidm/2157789184`. Online: accessed 08 April 2020.

National Oceanic and Atmospheric Administration (NOAA) (2019). Species directory: Atlantic salmon - protected. `https://www.fisheries.noaa.gov/species/atlantic-salmon-protected#overview`. Accessed: 20 February 2020.

Norberg, C. (2002). Pressure distributions around a circular cylinder in cross-flow. pages 1 – 4. Monash University, Melbourne, Australia.

Patursson, Ø. (2008). Flow through and around fish farming nets.

Pollock, D. W. (2012). *User Guide for MODPATH Version 6 - A Particle-Tracking Model for MODFLOW*. United States Geological Survey.

Pollock, D. W. (2016). *User Guide for MODPATH Version 7 - A Particle-Tracking Model for MODFLOW*. United States Geological Survey.

PubMed (2019). Cell biology protocols: Moi, pfu, and tcid50. `https://www.sciencegateway.org/protocols/cellbio/cell/moipfu.htm`. Accessed: 01 September 2019.

Qviller, L., Kristoffersen, A. B., Lyngstad, T. M., and Lillehaug, A. (2020). Infectious salmon anemia and farm-level culling strategies. *Frontiers in Veterinary Science*, 6:481.

Salama, N. K. and Murray, A. G. (2013). A comparison of modelling approaches to assess the transmission of pathogens between Scottish fish farms: The role of hydrodynamics and site biomass. *Preventive Veterinary Medicine*, 108(4):285 – 293. Special Issue: SVEPM 2012 - The cutting-edge of animal disease control in a global environment.

Salama, N. K. G. and Murray, A. G. (2011). Farm size as a factor in hydrodynamic transmission of pathogens in aquaculture fish production. *Aquaculture Environment Interactions*, 2:61 – 74.

Scanlon, T. (2019). marineFoam : Porous media modelling for flow through fish farm cages.

Spickler, A. R. (2010). Infectious salmon anemia. *Iowa State University: College of Veterinary Medicine*.

Toral, R. (2014). Introduction to the stochastic process. Institute for Cross-Disiplinary Physocs and Complex systems.

Trilling, D. (2017). Farmed versus wild salmon: Research review. `https://journalistsresource.org/studies/environment/food-agriculture/farmed-versus-wild-salmon-research-explainer/`. Accessed: 19 February 2020.

Turner, A. A., Jeans, T. L., and Reid, G. K. (2015). Experimental investigation of fish farm hydrodynamic wake properties on 1:15 scale model circular aquaculture cages. *Proceedings of the ASME 2015 34th International Conference on Ocean, Offshore and Arctic Engineering*, 34.

United Nations (2019). *World Population Prospects 2019: Data Booklet*.

ViralZone (2015). Isa virus. `https://viralzone.expasy.org/95?outline=all_by_species`. Accessed: 15 August 2018.

Wasserman, S. (2016). Choosing the right turbulence model for your cfd simulation. `https://www.engineering.com/DesignSoftware/DesignSoftwareArticles/ArticleID/13743/Choosing-the-Right-Turbulence-Model-for-Your-CFD-Simulation`. Accessed: 03 July 2020.

Watershed Watch Salmon Society (2004). Sea lice and salmon: Elevating the dialogue on the farmed-wild salmon story. Technical report, Watershed Watch Salmon Society.

Winthereig-Rasmussen, H., Simonsen, K., and Patursson, Ø. (2016). Flow through fish farming sea cages: Comparing computational fluid dynamics simulations with scaled and full-scale experimental data. *Ocean Engineering*, 124:21 – 31.

Appendix A

Salmon industry statistics

Chapter 1 discussed the importance of the efficient protein with a low environmental impact. This appendix looks at the impact of the animal protein industry on the environment as reported by Global Salmon Initiative [2020b].

Table A.1 tabulates the various aspects of environmental impact of the different livestock compared to salmon. Table A.1 shows how low the environmental impact of farmed salmon is, in comparison to chicken, pork, beef and lamb.

Table A.1: The environmental impact of livestock and salmon farming.

	Salmon	**Chicken**	**Pig**	**Cattle**	**Sheep**
Global production	3.2	107.1	118.2	66.0	9.3
Carbon footprint	0.60	0.88	1.30	5.92	-
Land use	3.7	7.1	11	102	185
Feed conversion ratio	1.2 to 1.5	1.7 to 2	2.7 to 5	6 to 10	-
Protein retention	28	37	21	14	-
Calorie retention	25	27	16	7	-
Edible yield	68	46	52	-	38

The rows in Table A.1 are explained below.

- Global production is the global production of farmed livestock and salmon measured in metric tonnes. The values in the table are millions.
- The carbon footprint is the measure of total greenhouse gasses emitted in the production of the product measured in grams of carbon dioxide (CO_2) per serving of product.
- The land use is the area of land occupied in square meters (m^2) per 100 g of protein produced. This also takes into account the land needed to feed the livestock and salmon.

- Feed conversion ratio is the ratio of food in kilograms needed to increase the mass of the animal by 1 kg in bodyweight. This measures how efficiently the animal makes use of resources.
- Protein retention is the gain in edible protein as a percentage of the protein consumed by the animal.
- Calorie retention is the gain in edible calories as a percentage of calories consumed by the animal.
- Edible yield is the percentage of the animal's total bodyweight that is fit for human consumption.

Appendix B

TRACE derivation

The derivation of the TRACE algorithm, that was discussed in Chapter 3, is expanded further in this appendix.

The TRACE algorithm begins with the Taylor expansion of the new particle position x^{t+1} around the old position x^t,

$$\begin{aligned} x^{k+1} &= x^k + \frac{dx}{dt}|_{x=x^k}\Delta t + \frac{1}{2}\frac{d^2x}{dt^2}\Delta t^2, \\ \frac{x^{k+1}-x^k}{\Delta t} &= \frac{dx}{dt} + \frac{1}{2}\frac{d^2x}{dt^2}\Delta t, \end{aligned} \tag{B.1}$$

where $\frac{dx}{dt}|_{x=x^k} = u(x^k, y^k)$,

$$\begin{aligned} \frac{x^{k+1}-x^k}{\Delta t} &= u(x^k, y^k) + \frac{1}{2}\frac{d}{dt}\left(\frac{dx}{dt}\right)\Delta t, \\ \frac{x^{k+1}-x^k}{\Delta t} &= u(x^k, y^k) + \frac{1}{2}\frac{du}{dt}\Delta t. \end{aligned}$$

The derivative of the velocity, $\frac{du}{dt}$ may then be expanded with the chain rule,

$$\frac{x^{k+1}-x^k}{\Delta t} = u(x^k, y^k) + \frac{1}{2}\left(\frac{\partial u}{\partial x}\frac{dx}{dt} + \frac{\partial u}{\partial y}\frac{dy}{dt}\right)\Delta t.$$

The derivatives of the positions, $\frac{dx}{dt}$ and $\frac{dy}{dt}$ may then be discretised between x^t and x^{t+1},

$$\begin{aligned} \frac{x^{k+1}-x^k}{\Delta t} &= u(x^k, y^k) + \frac{1}{2}\left(\frac{\partial u}{\partial x}\frac{\left(x^{k+1}-x^k\right)}{\Delta t} + \frac{\partial u}{\partial y}\frac{\left(y^{k+1}-y^k\right)}{\Delta t}\right)\Delta t, \\ \frac{x^{k+1}-x^k}{\Delta t} &= u(x^k, y^k) + \frac{1}{2}\frac{\partial u}{\partial x}\left(x^{k+1}-x^k\right) + \frac{1}{2}\frac{\partial u}{\partial y}\left(y^{k+1}-y^k\right). \end{aligned} \tag{B.2}$$

The equivalent equation may be derived in the y direction,

$$\frac{y^{k+1}-y^k}{\Delta t} = v(x^k, y^k) + \frac{1}{2}\frac{\partial v}{\partial x}\left(x^{k+1}-x^k\right) + \frac{1}{2}\frac{\partial v}{\partial y}\left(y^{k+1}-y^k\right). \tag{B.3}$$

These equations may be solved simultaneously for x^{k+1} and y^{k+1}. This process may be simplified by making the following substitutions for the constants in equations (B.2) and (B.3),

$$h_1 = 1 - \frac{\Delta t}{2}\frac{\partial u}{\partial x}, \tag{B.4a}$$
$$h_2 = u(x^k, y^k)\Delta t, \tag{B.4b}$$
$$h_3 = \left(1 - \frac{\Delta t}{2}\frac{\partial u}{\partial x}\right)x^k, \tag{B.4c}$$
$$h_4 = \frac{\Delta t}{2}\frac{\partial u}{\partial y}, \tag{B.4d}$$
$$h_5 = -\left(\frac{\Delta t}{2}\frac{\partial u}{\partial y}\right)y^k, \tag{B.4e}$$
$$h_6 = h_2 + h_3 + h_5, \tag{B.4f}$$

$$l_1 = 1 - \frac{\Delta t}{2}\frac{\partial v}{\partial y}, \tag{B.4g}$$
$$l_2 = v(x^k, y^k)\Delta t, \tag{B.4h}$$
$$l_3 = \frac{\Delta t}{2}\frac{\partial v}{\partial x}, \tag{B.4i}$$
$$l_4 = \left(-\frac{\Delta t}{2}\frac{\partial v}{\partial x}\right)x^k, \tag{B.4j}$$
$$l_5 = \left(1 - \frac{\Delta t}{2}\frac{\partial v}{\partial y}\right)y^k, \tag{B.4k}$$
$$l_6 = l_2 + l_4 + l_5. \tag{B.4l}$$

Substituting equations (B.4a) to (B.4l) into equation (B.2) and equation (B.3), results in the following two equations,

$$h_1 x^{k+1} = h_6 + h_4 y^{k+1}, \tag{B.5}$$
$$l_1 y^{k+1} = l_6 + l_3 x^{k+1}. \tag{B.6}$$

Rearranging equations (B.5) and (B.6), such that the x and y terms are separate, leads to the two equations,

$$x^{k+1} = \frac{h_6}{h_1} + \frac{h_4 l_6}{h_1 l_1 - l_3 h_6 - l_3 h_4}, \tag{B.7}$$
$$y^{k+1} = \frac{l_6}{l_1 - l_3 h_6 h_1^{-1} - l_3 h_4 h_1^{-1}}. \tag{B.8}$$

The u, v terms, and their partial derivatives, are bilinearly interpolated spatially to yield the following equations,

$$\frac{\partial u}{\partial x} = n\left(\frac{m\,u_{i,j} - (1-m)u_{i+1,j}}{\Delta x}\right) + (1-n)\left(\frac{m\,u_{i,j+1} - (1-m)u_{i+1,j+1}}{\Delta x}\right), \tag{B.9a}$$

$$\frac{\partial u}{\partial y} = m\left(\frac{n\,u_{i,j} - (1-n)u_{i,j+1}}{\Delta y}\right) + (1-m)\left(\frac{nu_{i+1,j} - (1-n)u_{i+1,j+1}}{\Delta y}\right), \tag{B.9b}$$

$$\frac{\partial v}{\partial x} = n\left(\frac{m\,v_{i,j} - (1-m)v_{i+1,j}}{\Delta x}\right) + (1-n)\left(\frac{m\,v_{i,j+1} - (1-m)v_{i+1,j+1}}{\Delta x}\right), \tag{B.9c}$$

$$\frac{\partial v}{\partial y} = m\left(\frac{n\,v_{i,j} - (1-n)v_{i,j+1}}{\Delta y}\right) + (1-m)\left(\frac{n\,v_{i+1,j} - (1-n)v_{i+1,j+1}}{\Delta y}\right), \tag{B.9d}$$

$$v = m\,n\,v_{i,j} + (1-m)n\,v_{i+1,j} + (1-m)(1-n)v_{i+1,j+1} + m\,(1-n)v_{i,j+1}, \tag{B.9e}$$

$$u = m\,n\,u_{i,j} + (1-m)n\,u_{i+1,j} + (1-m)(1-n)u_{i+1,j+1} + m\,(1-n)u_{i,j+1}, \tag{B.9f}$$

Solving for x^{k+1} and y^{k+1}, requires equations (B.4a) to (B.4l), as well as equations (B.9a) to (B.9f) to be substituted back into the two equations (B.7) and (B.8) for the new particle location. The solution to this is not as trivial as the literature would suggest. The solution will not be shown in this study.

Appendix C

ARIANE algorithm

The ARIANE algorithm was discussed by Blanke and Raynaud [1997]. This algorithm is similar to Pollock [2016], however this algorithm makes use of scaling factors, e_1, e_2 and e_3 in the three directions, and transport functions that are the velocity fields that are scaled with the above mentioned scale factors.

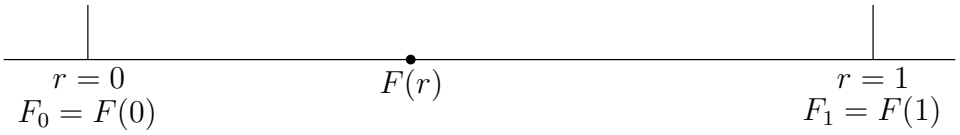


Figure C.1: A figure representing the transport within the cell in the x direction.

The algorithm makes use of a pseudo time, $s = (e_1\, e_2\, e_3)^{-1}t$ and the fractional position within the cell is defined as r where $r = 0$ at the one end of the cell and $r = 1$ at the other, which is illustrated in Figure C.1, where $r = xe_1$ in the x direction, such that the position r is fractional within the cell. These definitions of time and position are then used to define the transport through the cell as,

$$\frac{dr}{ds} = F. \tag{C.1}$$

Blanke and Raynaud [1997] make use of the non-divergence of the velocity within a cell. The non-divergence can then be written as,

$$\frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} + \frac{\partial H}{\partial z} = 0, \tag{C.2}$$

where F, G, H are the transports in the three directions with $F = e_2\, e_3 U$, $G = e_1\, e_3 V$ and $H = e_1\, e_2 W$. The terms U, V, W are the velocity fields where the components are the velocities in the x, y and z directions. The transports in all three directions are linearly dependent on position [Blanke and Raynaud, 1997].

The transport F as a function of the fractional position r can be written as,

$$F(r) = F_0 + r\Delta F, \tag{C.3}$$

where $r \in [0, 1]$, ΔF is the change in transport over the cell, i.e. $\Delta F = F(1) - F(0)$, $F(0) = F_0$ is transport at the inlet of the cell, i.e. $r = 0$.

The transport equation as a function of the change in pseudo-time and position (i.e. $U = dx/dt$) can be written as follows,

$$\frac{dr}{ds} = F, \tag{C.4}$$

where $s = (e_1 e_2 e_3)^{-1} t$. Equating equation (C.3) to equation (C.4) and integrating the resulting equation, the following expression is obtained,

$$\begin{aligned}
\frac{dr}{ds} &= F_0 + r\Delta F, \\
\int_0^r \frac{1}{F_0 + \Delta F r} dr &= \int_0^s ds, \\
s|_0^s &= \frac{1}{\Delta F} \left(\ln(F_0 + \Delta F r)|_0^r \right), \\
s &= \frac{1}{\Delta F} \ln\left(\frac{F_0 + \Delta F r}{F_0} \right), \qquad \text{(C.5)} \\
r &= \frac{F_0}{\Delta F} \left[e^{\Delta F s} - 1 \right]. \qquad \text{(C.6)}
\end{aligned}$$

If the transport is constant over the cell, i.e. $\Delta F = 0$ then the limit as $\Delta F \to 0$,

$$r = F_0 s, \tag{C.7}$$

and equivalent equations exist in each direction, for the transports G, and H.

For the ARIANE algorithm, the time it would take the particle to exit at each direction must be calculated, the shortest time will then indicate the exit direction.

The exit time for each direction can be calculated from equation C.5, i.e. when $r = 1$, for each direction respectively. The psudo time s, is the scaled time it takes for the particle to travel through the scaled cell, and thus,

$$s_1 - s_0 = \Delta s = \frac{1}{\Delta F} \ln\left(\frac{F_0 + \Delta F}{F_0} \right),$$

where s_0 is the pseudo time at $r = 0$, and s_1 is the pseudo time at $r = 1$, let $F(1) = F_1 = F_0 + \Delta F$,

$$s = \frac{1}{\Delta F} \ln\left(\frac{F_1}{F_0} \right). \tag{C.8}$$

When the transport does not change over the cell, i.e. $F_1 = F_0$, the limit $\Delta F \to 0$ is calculated as,

$$\Delta s = \frac{1}{F_0}. \tag{C.9}$$

As previously defined, the travelling time is then taken to be the shortest Δs over each of the three regions. This method is computationally expensive and limited by the same arguments as the MODPATH algorithm discussed in Section 3.3.1.

Appendix D

Vortex shedding

As is the case in Chapter 4, vortex shedding is experienced behind the cages. This appendix will look into why this phenomenon occurs.

There are three pressure gradient cases within a boundary layer that need to be considered. In the first case, in Figure D.1 a, there is no pressure gradient in the x direction, ie $\frac{\partial P}{\partial x} = 0$. The resulting velocity profile at a boundary is parabolic. In this case the fluid has constant velocity and according to Newton's first law, will continue to flow in the direction until acted upon by an external force.

The second case is depicted in Figure D.1 b. The pressure gradient is negative, $\frac{\partial P}{\partial x} \neq 0$, and fluid will travel from a high pressure to a low pressure.

In the final case, the pressure gradient is positive. The force, due to the pressure gradient, opposes the velocity of the fluid and the the fluid decelerates, and begins to flow in the negative direction. This velocity profile is seen in Figure D.1 c. The positive pressure gradient within this boundary layer is referred to as an adverse pressure gradient. The velocity profile has two points in which the velocity is zero. The first zero velocity point is at the boundary, such that the no slip condition is satisfied. The second zero velocity point is at the point at which a separation layer occurs. The stages of the development of the boundary layer and separation layer with an adverse pressure gradient are shown in Figure D.2.

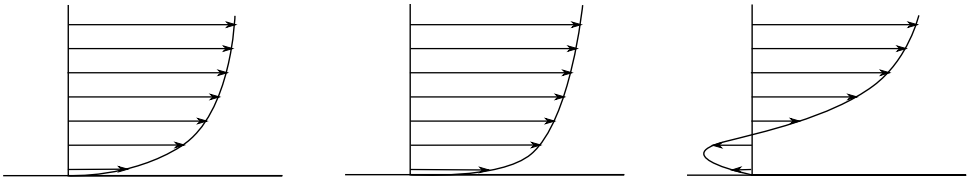


Figure D.1: Various pressure gradients showing different velocity profiles within a boundary layer.

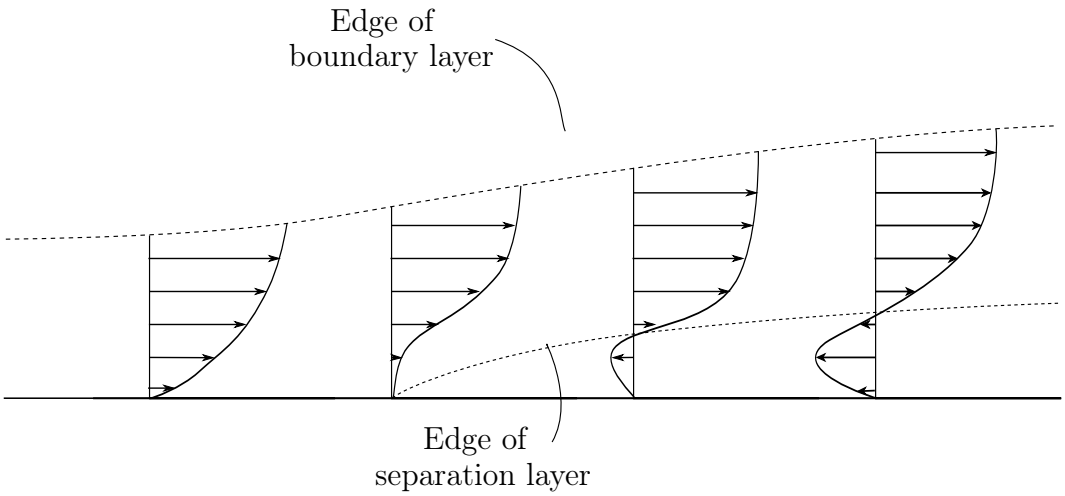


Figure D.2: The development of the velocity profiles within a boundary and separation layers.

According to experimental results in Norberg [2002], in Figure D.3 the pressure gradient between points a and b is negative, but the pressure gradient between b and c is positive. Due to the adverse pressure gradient, the flow profile is similar to that mentioned earlier, and a separation boundary forms. The separation of flow will form a von Kármán vortex street behind the cylinder.

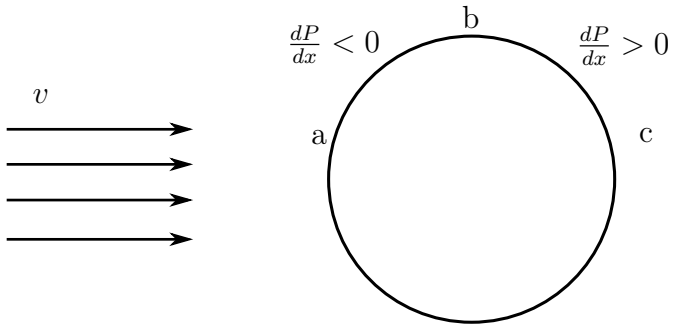


Figure D.3: Various pressure gradients around a sphere in a constant velocity field.

It should be noted that separation of flow is not due to turbulence. Turbulence is characterised as flow that is not predictable or periodic; von Kármán streets are predictable and periodic.

The vortex shedding behind cylindrical fish farms can be seen in the experimental results of Turner et al. [2015], but not in the computational results of Winthereig-Rasmussen et al. [2016] and Bi and Xu [2018]. When a turbulence model is used in the CFD simulation, these vortex streets are not observed. This study aims to determine whether or not these vortex streets make a considerable difference to the transfer of disease between the cages.